

HSD-TR-89-010



NOISE AND SONIC BOOM IMPACT TECHNOLOGY

Initial Development of an Assessment System for Aircraft Noise (ASAN): Software Listing

Volume IV of IV Volumes

**Sanford Fidell
Nicolaas Reddingius
Michael Harris
B. Andrew Kugler**

**BBN Systems & Technologies Corporation
21120 Vanowen Street
Canoga Park, CA 91303**

June 1989

Final Report for Period February 1987 - October 1988

Approved for public release; distribution is unlimited.

**Noise and Sonic Boom Impact Technology Program
Human Systems Division
Air Force Systems Command
Brooks Air Force Base, TX 78235-5000**

DTIC QUALITY INSPECTED 1

19980925 000

NOTICES

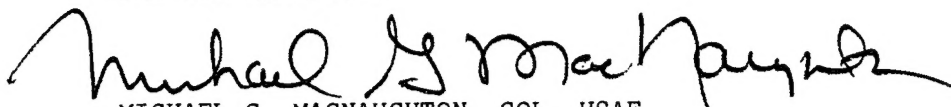
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility nor any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise as in any manner construed, as licensing the holder, or any other person or corporation; or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Office of Public Affairs has reviewed this report and it is releasable to the National Technical Information Service (NTIS), where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

ROBERT C. KULL, JR, Capt, USAF
NSBIT Program Manager

FOR THE COMMANDER



MICHAEL G. MACNAUGHTON, COL, USAF
Deputy Commander Development & Acquisition

Please do not request copies of this report from the Human Systems Division. Copies may be obtained from DTIC. Address your request for additional copies to:

Defense Technical Information Center
Cameron Station
Alexandria VA 22301-6145

If your address has changed, if you wish to be removed from our mailing list, or if your organization no longer employs the addressee, please notify HSD/SORT, Brooks AFB TX 78235-5000, to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Project 04515, Report 6624, NSBIT Task Order 0003			5. MONITORING ORGANIZATION REPORT NUMBER(S) HSD-TR-89-010 Vol. IV		
6a. NAME OF PERFORMING ORGANIZATION BBN Systems & Technologies Corporation		6b. OFFICE SYMBOL (If applicable) HSD/YA-NSBIT	7a. NAME OF MONITORING ORGANIZATION Advanced Development Program Office HSD/YA-NSBIT		
6c. ADDRESS (City, State, and ZIP Code) 21120 Vanowen Street Canoga Park, CA 91303			7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6573		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Advanced Development Program Office		8b. OFFICE SYMBOL (If applicable) HSD/YA-NSBIT	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-86-C-0530		
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6573			10. SOURCE OF FUNDING NUMBERS		
	PROGRAM ELEMENT NO. 63723F	PROJECT NO. 3037	TASK NO. 02	WORK UNIT ACCESSION NO. 01	
11. TITLE (Include Security Classification) Initial Development of an Assessment System for Aircraft Noise (ASAN), Volume IV: Source Code Listings					
12. PERSONAL AUTHOR(S) Fidell, Sanford; Reddingius, Nicolaas; Harris, Michael, and Kugler, B. Andrew					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM 2/12/87 TO 7/31/89		14. DATE OF REPORT (Year, Month, Day)	
				15. PAGE COUNT 276	
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP			
12	07		source code computer program ASAN		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This is the fourth volume of a volume report summarizing the development and current contents of a preliminary prototype version of an Assessment System for Aircraft Noise (ASAN). ASAN is a computer-based system intended to assist members of the United States Air Force (USAF) environmental planning community in addressing noise-related issues in developing environmental impact analysis documents, in compliance with USAF and other regulations, especially the National Environmental Policy Act (NEPA) of 1969. This volume contains technical appendices and listings of the source code for the preliminary prototype version of ASAN.					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL Robert C. Kull, Captain USAF			22b. TELEPHONE (Include Area Code) (513) 255-3384		22c. OFFICE SYMBOL HSD/YA-NSBIT

Table of Contents

APPENDIX A. SCREEN DESCRIPTION FILES	1
A.1 Screen Description File for Introductory Portions of ASAN	1
A.2 Screen Description File for MTR-related Operations	24
A.3 Screen Description File for Report-Related Portions of ASAN	102
A.4 Screen Description File for Graphic Portion of ASAN	106
APPENDIX B. PROGRAM LISTINGS	117
B.1 C Language Source Code	252

Appendix A

SCREEN DESCRIPTION FILES

This appendix contains the screen description files from which the U parser produces C code that can be compiled and linked with other object modules to produce an executable image. These files describe the appearance of the user interaction screens, and also determine what action the program takes when users elect specific actions.

A.1 Screen Description File for Introductory Portions of ASAN

```
-----
- INCLUDE STATEMENT FOR THE ASAN typedef DEFINITIONS NEEDED TO
-   COMPILE USERS.C WITHOUT INCURRING THE WRATH OF THE COMPILER
-----
```

```
INCLUDE ASANTYPE.H
INCLUDE ASAN.H
-----
```

```
-----
-DECLARATIONS FOR INTRODUCTORY SCREENS
-----
```

```
TEXTBLOCK introtxt2 {
    num_rows      8
    num_columns   76
    filename      txbk1/introtxt.txt
}

WINDOW introwindow1 {
    num_rows      10
    num_columns   78
    textblock     introtxt2, 1, 1
}

VARIABLE animals {
    type          INTEGER
    format        %d
}
DATUM animals {
    num_rows      1
    num_columns   30
    variable      animals, 0, 1
    pickable      NO
    trailer       "animals remain in list"
}

VARIABLE qual_entries {
    type          INTEGER
    format        %d
}
DATUM qualif {
    num_rows      1
    num_columns   32
    variable      qual_entries, 0, 10
    pickable      NO
    leader        "Thus far"
    trailer       "entries qualify"
}
DATUM sumofcit {
    num_rows      1
    num_columns   33
    variable      qual_entries, 0, 0
    trailer       "citations meeting criteria"
```

```

        pickable    NO
    }

VARIABLE curruit {
    type    INTEGER
    format %5d
}
DAFORM curruit {
    num_rows    1
    num_columns 31
    variable    curruit, 0 , 22
    leader      "Display shows number"
    trailer     "of"
    pickable    NO
}

VARIABLE when {
    foundin "VARCHAR dntep.arr"
    type    STRING
    format %4s
}
DAFORM when {
    num_rows    1
    num_columns 16
    variable    when, 0 , 11
    leader      "Published"
    pickable    NO
}

TEXTLINE canceltxt {"CANCEL SEARCH"}

BUTTON cancel {
    num_rows    1
    num_columns 17
    textline    canceltxt, 0,2
    helpfile    help/cancelrh.hlp
}

TEXTLINE contacttxt {"Interrogate point-of-contact database"}

BUTTON contacts {
    num_rows    1
    num_columns 40
    textline    contacttxt, 0,2
    helpfile    help/contacts.hlp
}

TEXTLINE humantxt {"Interrogate human effects citation index"}

BUTTON human {
    num_rows    1
    num_columns 50
    textline    humantxt, 0,2
    helpfile    help/humcit.hlp
}

TEXTLINE animaltxt {"Interrogate animal effects citation index"}

BUTTON animal {
    num_rows    1
    num_columns 50
    textline    animaltxt, 0,2
    helpfile    help/animcit.hlp
}

TEXTLINE structtxt {"Interrogate structural effects citation index"}

BUTTON struct {
    num_rows    1
    num_columns 50
    textline    structtxt, 0,2
    helpfile    help/structcit.hlp
}

TEXTLINE sonictxt {"Interrogate noise & sonic boom modeling citation index"}

BUTTON sonic {
    num_rows    1
    num_columns 60
    textline    sonictxt, 0,2
}

```

```

    helpfile    help/sonic.hlp
}

TEXTLINE legislatxt {"Interrogate legislative database"}

BUTTON legislat {
    num_rows    1
    num_columns 40
    textline    legislatxt, 0,2
    helpfile    help/legislat.hlp
}

TEXTLINE selectdbtxt {
    "You can now interrogate these ASEAN databases:"}

WINDOW inquiry {
    num_rows    10
    num_columns 78
    line        0,0, 0,77
    textline    selectdbtxt, 1,18
    button      contacts, 3,10, "CALL qsetup",
                                "NEW_SCREEN contactscreen"
    button      human, 4,10, "CALL qsetup",
                                "NEW_SCREEN haitisorn"
    button      animal, 5,10, "CALL qsetup",
                                "NEW_SCREEN aditisorn"
    button      struct, 6,10, "CALL qsetup",
                                "NEW_SCREEN scitisorn"
    button      sonic, 7,10, "CALL qsetup",
                                "NEW_SCREEN maitisorn"
    button      legislat, 8,10, "NEW_SCREEN legislatcreen"
}

SCREEN dbinq {
    mainscreen YES
    border YES
    window    inquiry, 10,1
    - window  majoraction4, 19,1
    window    introwindow1, 1, 1
}

-----
-           Declarations for POINT OF CONTACT DATABASE SCREEN
-----

TEXTLINE stsearchtxt {"START SEARCH"}

BUTTON stsearch {
    num_rows    1
    num_columns 15
    textline    stsearchtxt, 0,2
    helpfile    help/stsearch.hlp
}

TEXTLINE caff014txt {"AFF-014 attributes"}

BUTTON caff014 {
    num_rows    1
    num_columns 20
    textline    caff014txt, 0,2
    helpfile    help/caff014.hlp
}

TEXTLINE ctribaltxt {"Tribal"}

BUTTON ctribal {
    num_rows    1
    num_columns 9
    textline    ctribaltxt, 0,2
    helpfile    help/ctribal.hlp
}

TEXTLINE cmilitarytxt {"Military"}

BUTTON cmilitary {
    num_rows    1
    num_columns 11
    textline    cmilitarytxt, 0,2
    helpfile    help/cmilitar.hlp
}

```

```

    )

TEXTLINE cfederaltxt ("Federal")

BUTTON cfederal {
    num_rows 1
    num_columns 10
    textline cfederaltxt, 0, 2
    helpfile help/cfederal.hlp
}

TEXTLINE cstatetxt ("State")

BUTTON cstate {
    num_rows 1
    num_columns 8
    textline cstatetxt, 0, 2
    helpfile help/cstate.hlp
}

TEXTLINE ccountytxt ("County")

BUTTON ccounty {
    num_rows 1
    num_columns 9
    textline ccountytxt, 0, 2
    helpfile help/ccounty.hlp
}

TEXTLINE ccitytxt ("City")

BUTTON ccity {
    num_rows 1
    num_columns 7
    textline ccitytxt, 0, 2
    helpfile help/ccity.hlp
}

VARIABLE address {
    type STRING
    format 4-25s
}

DATUM address {
    num_rows 1
    num_columns 77
    variable address, 0, 28
    leader "City and/or state address: "
    helpfile help/address.hlp
}

VARIABLE contact {
    type STRING
    format 4-15s
}

DATUM contact {
    num_rows 1
    num_columns 77
    variable contact, 0, 28
    leader "Point of contact last name:"
    helpfile help/contact.hlp
}

VARIABLE affselec {
    type STRING
    format 4-15s
}

DATUM affselec {
    num_rows 1
    num_columns 50
    variable affselec, 0, 33
    leader "Affiliation currently selected:"
    pickable NO
}

TEXTLINE agenttxt ("Agency:")

TEXTLINE searchtxt ("Search by:")

WINDOW contactsearch {
    num_rows 17
    num_columns 78

```

```

line      0,0, 0,77
textline  searchtxt, 1, 1
datum     countam, 3, 1
datum     address, 4, 1
textline  agenttxt, 5, 1
button    city, 5,10, "CALL staff 0"
button    county, 5,10, "CALL staff 1"
button    estate, 5,27, "CALL staff 2"
button    cfederal, 5,37, "CALL staff 3"
button    cmilitary, 5,48, "CALL staff 4"
button    ctribal, 5,60, "CALL staff 5"
datum     affsaled, 6, 1
button    caffs14, 9, 1, "NEW_SCREEN countsearch"
button    stsearch, 16, 1, "CALL pscrch"
button    cnsrch, 16,48, "CALL qscrch",
                        "NEW_SCREEN dbinq"
}

```

```

SCREEN contactscreen {
  title      "POINT OF CONTACT DATABASE"
  window     contactsearch, 2,1
- window     majoraction4, 19,1
  border     YES
}

```

```

-----
- Point of contact attribute search screen
-----

```

```
TEXTLINE minoratxt ("Minor attributes")
```

```
TEXTLINE scrolltxt ("Scroll window for more alternatives")
```

```
TEXTBLOCK minorat {
  filename   txtblk/minor.txt
  num_rows   10
  num_columns 35
  border     YES
}

```

```
WINDOW minorat {
  num_rows   10
  num_columns 35
  textline   minoratxt, 1,0
  textblock  minorat, 2,0
  textline   scrolltxt, 12,0
}

```

```
TEXTBLOCK majorat {
  filename   txtblk/major.txt
  num_rows   10
  num_columns 35
  border     YES
}

```

```
VARIABLE minorat {
  type  STRING
  format 4-45s
}

```

```
DATUM minorat {
  num_rows 1
  num_columns 77
  variable minorat, 0,22
  loader "Type minor attribute:"
  helpfile help/minor.hlp
}

```

```
VARIABLE majorat {
  type  STRING
  format 4-15s
}

```

```
DATUM majorat {
  num_rows 1
  num_columns 77
  variable majorat, 0,22
  loader "Type major attribute:"
  helpfile help/major.hlp
}

```

```

)

TEXTLINE majorstxt {"Major attributes"}

WINDOW contactsearch {
    num_rows      17
    num_columns   78
    line          0,0, 0,77
    datum         majorst, 1, 1 -, "ADD_WINDOW minorst 5 41"
    datum         minorst, 2, 1 -, "CALL dummy"
    textline      majorstxt, 4, 1
    textline      scrolltxt, 15, 1
    textblock     majorst, 5, 1
    textblock     minorst, 5,41
    button        stsearch, 16, 1, - "REMOVE_WINDOW"
                                "CALL psrch"
    button        enasrch, 16,45, "CALL qsetup",
                                "NEW_SCREEN doing"
}

SCREEN contactsearch {
    title         "POINT OF CONTACT ATTRIBUTE SEARCH"
    window        contactsearch, 2,1
    - window      majorstxt4, 19,1
    border        YES
}

-----
-           Point of contact display screen
-
-*****ERR: all of these poc variables need to be declared in a header file
-----

VARIABLE F_NAME {
    foundin      "VARCHAR f_name.arr"
    type         STRING
    format       4-10s
}

DATUM contactname1 {
    num_rows     1
    num_columns   25
    variable      F_NAME, 0, 15
    leader        "Contact Name:"
    pickable      NO
}

VARIABLE L_NAME {
    foundin      "VARCHAR l_name.arr"
    type         STRING
    format       4-15s
}

DATUM contactname2 {
    num_rows     1
    num_columns   30
    variable      L_NAME, 0, 1
    pickable      NO
}

VARIABLE CORTITLE { -TITLE may be a reserved word
    foundin      "VARCHAR cortitle.arr"
    type         STRING
    format       4-45s
}

DATUM cortitle {
    num_rows     1
    num_columns   70
    variable      CORTITLE, 0, 9
    leader        "Title:"
    pickable      NO
}

VARIABLE OFFICE {
    foundin      "VARCHAR office.arr"
    type         STRING

```

```

format          4-45s
)

DATUM office (
  num_rows      1
  num_columns    70
  variable       OFFICE, 0, 15
  leader         "Office:"
  pickable      NO
)

VARIABLE AGENCY_DEPT (
  foundin "VARCHAR agency_dept.arr"
  type    STRING
  format  4-45s
)

DATUM agency (
  num_rows      1
  num_columns    70
  variable       AGENCY_DEPT, 0, 10
  leader         "Agency:"
  pickable      NO
)

VARIABLE ST_ADD_DIV (  ***only 25 characters for a street address?
  foundin "VARCHAR st_add_div.arr"
  type    STRING
  format  4-25s
)

DATUM st_add_div (
  num_rows      1
  num_columns    50
  variable       ST_ADD_DIV, 0, 17
  leader         "Street Address:"
  pickable      NO
)

VARIABLE PO_BOX (
  foundin "VARCHAR po_box.arr"
  type    STRING
  format  4-10s
)

DATUM po_box (
  num_rows      1
  num_columns    40
  variable       PO_BOX, 0, 10
  leader         "Post Office Box:"
  pickable      NO
)

VARIABLE MISC_ADD (
  foundin "VARCHAR misc_add.arr"
  type    STRING
  format  4-10s
)

DATUM misc_add (
  num_rows      1
  num_columns    20
  variable       MISC_ADD, 0, 2
  pickable      NO
)

VARIABLE CITY_BASE (
  foundin "VARCHAR city_base.arr"
  type    STRING
  format  4-25s
)

DATUM city_base (
  num_rows      1
  num_columns    30
  variable       CITY_BASE, 0, 1
  pickable      NO
)

VARIABLE STATE (
  foundin "VARCHAR state.arr"
  type    STRING

```

```

        format          4-2s
    }

    DATUM state {
        num_rows        1
        num_columns      5
        variable         STATE, 0, 1
        pickable         NO
    }

    VARIABLE ZIPCODES {
        foundia "VARCHAR zipcode.arr"
        type    STRING
        format  4-9s
    }

    DATUM zipcode {
        num_rows        1
        num_columns      10
        variable         ZIPCODES, 0, 1
        pickable         NO
    }

    VARIABLE MAIL_CODE {
        foundia "VARCHAR mail_code.arr"
        type    STRING
        format  4-10s
    }

    DATUM mail_code {
        num_rows        1
        num_columns      40
        variable         MAIL_CODE, 0, 12
        leader           "Mail Code:"
        pickable         NO
    }

    VARIABLE PHONE {
        foundia "VARCHAR phone.arr"
        type    STRING
        format  4-20s
    }

    DATUM phone {
        num_rows        1
        num_columns      40
        variable         PHONE, 0, 20
        leader           "Telephone Number:"
        pickable         NO
    }

    VARIABLE AFFILIATIO {
        foundia "VARCHAR affiliatio.arr"
        type    STRING
        format  4-8s
    }

    DATUM affiliatio {
        num_rows        1
        num_columns      30
        variable         AFFILIATIO, 0, 17
        leader           "Affiliation:"
        pickable         NO
    }

    VARIABLE MAJOR_ATTRIB {
        foundia "VARCHAR major_attrib.arr"
        type    STRING
        format  4-16s
    }

    DATUM major_attribute {
        num_rows        1
        num_columns      35
        variable         MAJOR_ATTRIB, 0, 19
        leader           "Major Attribute:"
        pickable         NO
    }

    VARIABLE MINOR_ATTRIBUTE {
        foundia "VARCHAR minor_attribute.arr"
        type    STRING

```



```

format          %45s
}

DATUM minor_attribute {
  num_rows      1
  num_columns    70
  variable       MINOR_ATTRIBUTE, 0, 18
  leader         "Minor Attribute:"
  pickable       NO
}

VARIABLE AREA {
  foundin "VARCHAR area.arr"
  type     STRING
  format   %45s
}

DATUM area {
  num_rows      1
  num_columns    30
  variable       AREA, 0, 7
  leader         "AREA:"  ~(whatever that means)
  pickable       NO
}

VARIABLE SCOPE {
  foundin "VARCHAR scope.arr"
  type     STRING
  format   %45s
}

DATUM scope {
  num_rows      1
  num_columns    30
  variable       SCOPE, 0, 8
  leader         "Scope:"  ~(whatever that means)
  pickable       NO
}

```

```

WINDOW poststuff {
  num_rows      17
  num_columns    78
  line          0, 0, 0.77
  datum         contactname1,      1, 2
  datum         contactname2,      1, 40
  datum         contacttitle,      2, 2
  datum         office,            3, 2
  datum         agency,            4, 2
  datum         st_add_div,        5, 2
  datum         po_box,            6, 2
  datum         misc_add,          6, 25
  datum         city_base,        7, 2
  datum         state,            7, 25
  datum         zipcode,          7, 35
  datum         mail_code,        8, 2
  datum         phone,            9, 2
  datum         affiliatio,       9, 41
  datum         major_attribute,  10, 2
  datum         minor_attribute,  11, 2
  datum         area,            12, 2
  datum         scope,           12, 40
}

```

```

TEXTLINE nextaddr {"Show next address"}

```

```

BUTTON nextaddr {
  num_rows      1
  num_columns    25
  textline      nextaddr, 0, 2
  helpfile      help/nohelp.hlp
}

```

```

WINDOW getoffscreens {
  num_rows      4
  num_columns    78
  line          0, 0, 0.77
  button        nextaddr, 2, 1, "CALL nextpoc"
  button        stsearch, 2, 1, "CALL dummy"
  button        caasch, 2, 45, "CALL qsetup",

```

```

                                "NEW_SCREEN dialog"
)

SCREEN poeisplay (
    title      "DISPLAY POINT OF CONTACT INFORMATION"
    window     poestuff,      2,1
    window     getoffscreen,  18,1
    border     YES
)

-----
-      Human effects citation database screen
-----

VARIABLE titlfrag (
    foundin "VARCHAR titlfrag.arr"
    type     STRING
    format   4-60s
)
DATUM titlfrag (
    num_rows 1
    num_columns 65
    variable  titlfrag, 0,0
    helpfile  help/titlfrag.hlp
)

TEXTLINE keywrtdxt ("Keyword categories")

BUTTON keywrtd (
    num_rows 1
    num_columns 25
    textline  keywrtdxt, 0, 2
    helpfile  help/keywrtd.hlp
)

VARIABLE dates2 (
    type     INTEGER
    format   44d
)
DATUM dates2 (
    num_rows 1
    num_columns 15
    variable  dates2, 0,4
    leader    "and"
    trailer   "(year)"
    helpfile  help/date.hlp
)

VARIABLE dates1 (
    type     INTEGER
    format   44d
)
DATUM dates1 (
    num_rows 1
    num_columns 27
    variable  dates1, 0,14
    leader    "Data between:"
    trailer   "(year)"
    helpfile  help/date.hlp
)

VARIABLE authornam (
    foundin "VARCHAR authornam.arr"
    type     STRING
    format   4-40s
)
DATUM author (
    num_rows 1
    num_columns 77
    variable  authornam, 0,20
    leader    "Author's last name:"
    helpfile  help/author.hlp
)

VARIABLE citautnam1 (
    foundin "VARCHAR40 authorlist[0].arr"
    type     STRING
    format   4-30s
)
DATUM citautnam1 (
    num_rows 1
    num_columns 40

```

```

- Note: The string lengths
- are to get by the parser.
- The program will figure out
- whether to allow 30 or 40
- columns on the display.

```

```

        variable    citautnam1, 0, 9
        leader      "Author:"
        pickable    NO
    }
VARIABLE citautnam2 {
    foundin "VARCHAR40 authorlist[1].arr"
    type    STRING
    format  4-30s
}
DATUM citautnam2 {
    num_rows    1
    num_columns  40
    variable    citautnam2, 0, 9
    pickable    NO
}

VARIABLE citautnam3 {
    foundin "VARCHAR40 authorlist[2].arr"
    type    STRING
    format  4-30s
}
DATUM citautnam3 {
    num_rows    1
    num_columns  40
    variable    citautnam3, 0, 9
    pickable    NO
}

VARIABLE citautnam4 {
    foundin "VARCHAR40 authorlist[3].arr"
    type    STRING
    format  4-30s
}
DATUM citautnam4 {
    num_rows    1
    num_columns  40
    variable    citautnam4, 0, 9
    pickable    NO
}

VARIABLE citautnam5 {
    foundin "VARCHAR40 authorlist[4].arr"
    type    STRING
    format  4-30s
}
DATUM citautnam5 {
    num_rows    1
    num_columns  40
    variable    citautnam5, 0, 9
    pickable    NO
}

VARIABLE citautnam6 {
    foundin "VARCHAR40 authorlist[5].arr"
    type    STRING
    format  4-20s
}
DATUM citautnam6 {
    num_rows    1
    num_columns  23
    variable    citautnam6, 0, 0
    pickable    NO
}

VARIABLE citautnam7 {
    foundin "VARCHAR40 authorlist[6].arr"
    type    STRING
    format  4-20s
}
DATUM citautnam7 {
    num_rows    1
    num_columns  23
    variable    citautnam7, 0, 0
    pickable    NO
}

VARIABLE citautnam8 {
    foundin "VARCHAR40 authorlist[7].arr"
    type    STRING
    format  4-20s
}
DATUM citautnam8 {

```

```

num_rows      1
num_columns   23
variable      citautasm9, 0, 0
pickable     NO
}

VARIABLE citautasm9 {
  foundin "VARCHAR40 authorlist[8].arr"
  type    STRING
  format  4-20s
}
DATUM citautasm9 {
  num_rows      1
  num_columns   23
  variable      citautasm9, 0, 0
  pickable     NO
}

VARIABLE citautasm10 {
  foundin "VARCHAR40 authorlist[9].arr"
  type    STRING
  format  4-20s
}
DATUM citautasm10 {
  num_rows      1
  num_columns   23
  variable      citautasm10, 0, 0
  pickable     NO
}

TEXTLINE emtitltxt ("Title:")

WINDOW haitiscrn {
  num_rows      16
  num_columns   78
  line          0, 0, 0.77
  textline      searchtxt, 1, 1
  datum         qualif, 1, 45
  datum         author, 3, 1, "CALL VCAPITAL &authorasm",
                        "NEWVALS"
  textline      emtitltxt, 4, 1
  datum         titlefrag, 4, 9, "CALL VCAPITAL &titlefrag",
                        "NEWVALS"
  datum         dates1, 8, 1
  datum         dates2, 8, 27
  button        keywrd, 10, 1, "CALL hscrh010"
  button        stsearch, 15, 1, "CALL hscrh001"
  button        cnsasch, 15, 45, "CALL qsetup",
                        "NEW_SCREEN dbing"
}

SCREEN haitiscrn {
  title         "FUMAN EFFECTS CITATION SEARCH"
  window        haitiscrn, 2, 1
-  window        majoraction3, 10, 1
  border        YES
}

```

 - Declarations for animal effects citation database screen

```

WINDOW acitiscrn {
  num_rows      16
  num_columns   78
  line          0, 0, 0.77
  textline      searchtxt, 1, 1
  datum         qualif, 1, 45
  datum         author, 3, 1, "CALL VCAPITAL &authorasm",
                        "NEWVALS"
  textline      emtitltxt, 4, 1
  datum         titlefrag, 4, 9, "CALL VCAPITAL &titlefrag",
                        "NEWVALS"
  datum         dates1, 8, 1
  datum         dates2, 8, 27
  button        keywrd, 10, 1, "CALL asrch010"
  button        stsearch, 15, 1, "CALL asrch001"
  button        cnsasch, 15, 45, "CALL qsetup",
                        "NEW_SCREEN dbing"
}

```

```

SCREEN scitiscrn {
  title      "ANIMAL EFFECTS CITATION SEARCH"
  window     scitiscrn, 2,1
-  window     majoraction3, 18,1
  border     YES
}

```

```

- Declaration for structural effects citation database screen

```

```

WINDOW scitiscrn {
  num_rows    16
  num_columns  78
  line        0,0, 0,77
  textline    searchtxt, 1, 1
  datum       qualif, 1,45
  datum       author, 3, 1, "CALL VCAPITAL &authorname",
                                "NEWSVALS"
  textline    entitl1txt, 4, 1
  datum       titlefrag, 4, 9, "CALL VCAPITAL &titlefrag",
                                "NEWSVALS"
  datum       dates1, 8, 1
  datum       dates2, 8,27
  button      keywrd, 10, 1, ~"CALL serch010"
                                "CALL dummy"
  button      stsearch, 15, 1, "CALL serch001"
  button      cancsch, 15,45, "CALL qstetup",
                                "NEW_SCREEN dbing"
}

```

```

SCREEN scitiscrn {
  title      "STRUCTURAL EFFECTS CITATION SEARCH" --ar 2/4 Corrected
  window     scitiscrn, 2,1
-  window     majoraction3, 18,1
  border     YES
}

```

```

- Declaration for noise and sonic boom modeling effects citation database

```

```

TEXTLINE notimplement ("NOT IMPLEMENTED IN PROTOTYPE VERSION OF ASAN")

```

```

WINDOW scitiscrn {
  num_rows    16
  num_columns  78
  line        0, 0, 0, 77
  textline    searchtxt, 1, 1
  datum       qualif, 1,45
  datum       author, 3, 1, "CALL VCAPITAL &authorname",
                                "NEWSVALS"
  textline    entitl1txt, 4, 1
  datum       titlefrag, 4, 9, "CALL VCAPITAL &titlefrag",
                                "NEWSVALS"
  datum       dates1, 8, 1
  datum       dates2, 8,27
  button      keywrd, 10, 1, ~"CALL serch010"
                                "CALL dummy"
  button      stsearch, 15, 1, "CALL serch001"
  button      cancsch, 15,45, "CALL qstetup",
                                "NEW_SCREEN dbing"
}

```

```

SCREEN scitiscrn {
  title      "NOISE AND SONIC BOOM MODELING EFFECTS SEARCH"
  window     scitiscrn, 2,1
-  window     majoraction3, 18,1
  border     YES
}

```

```

- Declaration for LEGISLATIVE DATABASE SCREEN

```

```

WINDOW legislatscreen {
  num_rows    16

```

```

num_columns 78
line 0,0, 0,77
textline searchtxt, 1, 1
datum qualif, 1,30
datum author, 3, 1, "CALL VCAPITAL &authorname",
"NEWVALS"
textline entitltxt, 4, 1
datum titlefrag, 4, 9, "CALL VCAPITAL sttitlefrag",
"NEWVALS"
datum dates1, 8, 1
datum dates2, 8,27
button keywrd, 10, 1, "-NEW_SCREEN lkeywrdcscreen"
"CALL dummy"
- textline notimplment, 3,15
button stsearch, 15,1, "CALL dummy"
button canasech, 15,45, "CALL qsetup",
"NEW_SCREEN dbing"
)

```

```

SCREEN legislatscreen (
title "LEGISLATIVE CITATION DATABASE SEARCH"
window legislatscreen, 2,1
- window majoraction3, 16,1
border YES
)

```

Declaration for Animal effects keyword category search screen

```

VARIABLE study2 (
type STRING
format 4-20s
)
DATUM study2 (
num_rows 1
num_columns 37
variable study2, 0 ,14
leader "Study type 2:"
helpfile help/study2.hlp
)
DATUM study2d (
num_rows 1
num_columns 37
variable study2, 0 ,14
leader "Study type 2:"
pickable NO
)

```

```

VARIABLE study1 (
type STRING
format 4-20s
)
DATUM study1 (
num_rows 1
num_columns 37
variable study1, 0 ,14
leader "Study type 1:"
helpfile help/study1.hlp
)
DATUM study1d (
num_rows 1
num_columns 37
variable study1, 0 ,14
leader "Study type 1:"
pickable NO
)

```

```

VARIABLE expdesc (
foundin "VARCHAR expdesc.ans"
type STRING
format 4-20s
)
DATUM expdesc (
num_rows 1
num_columns 40
variable expdesc, 0 ,19
leader "Experimental type:"
helpfile help/method1.hlp
)
DATUM expdescd (

```

```

num_rows      1
num_columns   40
variable      expdesc, 0,19
leader        "Experimental type:"
pickable     NO
}

VARIABLE noistype {
  foundin "VARCHAR noistype.arr"
  type    STRING
  format  4-20s
}
DATUM noistype {
  num_rows      1
  num_columns   37
  variable      noistype, 0,12
  leader        "Noise type:"
  helpfile      help/noistype.hlp
}
DATUM noistyped {
  num_rows      1
  num_columns   37
  variable      noistype, 0,12
  leader        "Noise type:"
  pickable     NO
}

VARIABLE species4 {
  foundin "VARCHAR40 species4.arr"
  type    STRING
  format  4-40s
}
DATUM species4 {
  num_rows      1
  num_columns   65
  variable      species4, 0,16
  leader        "Species type 4:"
  helpfile      help/species.hlp
}
DATUM species4d {
  num_rows      1
  num_columns   65
  variable      species4, 0,16
  leader        "Species type 4:"
  pickable     NO
}

VARIABLE species3 {
  foundin "VARCHAR40 species3.arr"
  type    STRING
  format  4-40s
}
DATUM species3 {
  num_rows      1
  num_columns   65
  variable      species3, 0,16
  leader        "Species type 3:"
  helpfile      help/species.hlp
}
DATUM species3d {
  num_rows      1
  num_columns   65
  variable      species3, 0,16
  leader        "Species type 3:"
  pickable     NO
}

VARIABLE species2 {
  foundin "VARCHAR40 species2.arr"
  type    STRING
  format  4-40s
}
DATUM species2 {
  num_rows      1
  num_columns   65
  variable      species2, 0,16
  leader        "Species type 2:"
  helpfile      help/species.hlp
}
DATUM species2d {
  num_rows      1
  num_columns   65

```

```

        variable    species2, 0, 16
        leader      "Species type 2:"
        picktable   NO
    }

VARIABLES species1 {
    foundin "VARCHAR40 species1.arr"
    type    STRINGS
    format  4-40s
}
DATUM species1 {
    num_rows    1
    num_columns  65
    variable    species1, 0, 16
    leader      "Species type 1:"
    helpfile    help/species.hlp
}
DATUM species1d {
    num_rows    1
    num_columns  65
    variable    species1, 0, 16
    leader      "Species type 1:"
    picktable   NO
}

TEXTBLOCK animal {
    num_rows    8
    num_columns  35
    filename    txtblk/animal.txt
    border      YES
}

TEXTBLOCK var_animal {
    num_rows    8
    num_columns  45
    filename    txtblk/varanimal.txt
    border      YES
}

WINDOW spec1entry {
    num_rows    1
    num_columns  50
    datum      species2, 0, 1, "CALL VCAPITAL &species2",
                                "NEWVALS",
                                "CALL queryal2"
}

WINDOW spec3entry {
    num_rows    1
    num_columns  50
    datum      species3, 0, 1, "CALL VCAPITAL &species3",
                                "NEWVALS",
                                "CALL queryal3"
}

WINDOW spec4entry {
    num_rows    1
    num_columns  50
    datum      species4, 0, 1, "CALL VCAPITAL &species4",
                                "NEWVALS",
                                "CALL queryal4"
}

WINDOW akeywrdsearch {
    num_rows    19
    num_columns  78
    line        0,0, 0,77
    datum      qualify, 1, 1
    datum      animals, 1, 40
    datum      species1, 2, 1, "CALL VCAPITAL &species1",
                                "NEWVALS",
                                "CALL queryal1"

    datum      species2d, 3, 1
    datum      species3d, 4, 1
    datum      species4d, 5, 1
    datum      noistype, 6, 1
    datum      expdesed, 6, 38

```



```

        datum      study1d,    7, 1
        datum      study2d,    7,38
        textblock   animal,     8,23
        textline    scrolltxt,  16,23
        button      stsearch,   18, 1, "CALL asrch002"
        button      canrch,     18,55, "CALL qsetup",
                                "NEW_SCREEN dbing"
        helpfile    help/nohelp.hlp
    }

    SCREEN akeyrch {
        title       "ANIMAL EFFECTS KEYWORD SEARCH"
        window      akeyrchsearch, 2,1
        border      YES
    }

    WINDOW akeyalt { - alternate used for pop-up textblock
        num_rows    19
        num_columns  78
        line        0,0, 0,77
        datum      qualify,     1, 1
        datum      animals,     1,40
        datum      species1d,    2, 1
        datum      species2d,    3, 1
        datum      species3d,    4, 1
        datum      species4d,    5, 1
        datum      noistype,     6, 1, "CALL VCAPITAL facistype",
                                "NEWVALS",
                                "CALL dummy"
        datum      expdesc,     6,38, "CALL VCAPITAL expdesc",
                                "NEWVALS",
                                "CALL dummy"
        datum      study1,      7, 1, "CALL dummy"
        datum      study2,      7,38, "CALL dummy"
        textblock   var_animal,  8,18
        textline    scrolltxt,  16,23
        button      stsearch,   18, 1, "CALL asrch002"
        button      canrch,     18,55, "CALL qsetup",
                                "NEW_SCREEN dbing"
        helpfile    help/nohelp.hlp
    }

    SCREEN akeyalt {
        title       "ANIMAL EFFECTS KEYWORD SEARCH"
        window      akeyalt, 2,1
        border      YES
    }

    -----
    ~ Declaration for Human effects keyword category search screen
    -----

    TEXTBLOCK exptype {
        num_rows    10
        num_columns  35
        filename     txtblk/methodol.txt
        border      YES
    }

    WINDOW exptype {
        num_rows    10
        num_columns  35
        textblock    exptype, 0,0
    }

    TEXTBLOCK noistype {
        num_rows    10
        num_columns  35
        filename     txtblk/asetype.txt
        border      YES
    }

    WINDOW noistype {
        num_rows    10
        num_columns  35
        textblock    noistype, 0,0
    }

    TEXTBLOCK hexpdesc {
        num_rows    10
        num_columns  35
        filename     txtblk/humexp.txt
    }

```

```

border      YES
)

VARIABLE descotype {
  foundin "VARIABLE descotype.arr"
  type    STRING
  format  4-45s
}
DATUM descotype {
  num_rows      1
  num_columns    77
  variable      descotype, 0, 24
  leader        "Effect descriptor type:"
  helpfile      help/humimp.hlp
}
DATUM structeff {
  num_rows      1
  num_columns    77
  variable      descotype, 0, 22
  leader        "Structural effect on:"
  helpfile      help/strimpac.hlp
}

WINDOW hkeywordsearch {
  num_rows      19
  num_columns    78
  line          0, 0, 0, 77
  datum         qualif,      1, 30
  datum         descotype,   2, 1, "CALL VCAPITAL &descotype",
                                "NEWVALS",
                                "CALL hsrch011"
  textblock     hsfdesc,     6, 23
  datum         noistype,    3, 1, "CALL VCAPITAL &noistype",
                                "NEWVALS",
                                "CALL hsrch012"
  datum         expdesc,     4, 1, "CALL VCAPITAL &expdesc",
                                "NEWVALS",
                                "CALL hsrch013"
  textline      scrolltxt,   16, 23
  button        stsearch,    18, 1, "CALL hsrch002"
  button        cancsch,     18, 45, "CALL qsetup",
                                "NEW_SCREEN chinq"
}

SCREEN hkeyrch {
  title         "HUMAN EFFECTS KEYWORD SEARCH"
  window        hkeywordsearch, 2, 1
  - window      majoraction, 18, 1
  border        YES
}

-----
-   declaration for Structural effects keyword category search screen
-----

TEXTBLOCK sffdesc {
  num_rows      10
  num_columns    35
  filename      txtblk/strimpac.txt
  border        YES
}

WINDOW ekeywordsearch {
  num_rows      18
  num_columns    78
  line          0, 0, 0, 77
  datum         qualif,      1, 30
  datum         structeff,   1, 1, "CALL VCAPITAL &descotype",
                                "NEWVALS",
                                "ADD_WINDOW noistype 7 24"
  textblock     sffdesc,     8, 23
  datum         noistype,    2, 1, "REMOVE_WINDOW",
                                "CALL VCAPITAL &noistype",
                                "NEWVALS",
                                "ADD_WINDOW expstype 7 24",

```

```

        datum      expdesc,      3, 1, "CALL dummy"
                                "CALL VCAPITAL expdesc",
                                "NEWVALS",
                                "CALL dummy"
        textline    scrolltxt,    15,23
        button      stsearch,     17, 1, "CALL dummy"
        button      cansearch,    17,45, "CALL qsetup",
                                "NEW_SCREEN doing"
    }

    SCREEN skkeyrsh {
        title        "STRUCTURAL EFFECTS KEYWORD SEARCH"
        window       skkeyrdssearch, 2,1
        - window     majoraction3, 10,1
        border       YES
    }

-----
- Declaration for Noise & Sonic-boom effects keyword category search screen
-----

TEXTBLOCK abdesc {
    num_rows        10
    num_columns      35
    filename         txtblk/abimpac.txt
    border          YES
}

- VARIABLE noiseff {
-     type          STRINGS
-     format        4-45s
- }

- DATUM noiseff {
-     num_rows      1
-     num_columns    77
-     variable       noiseff, 0,32
-     leader         "Noise and sonic boom affect on:"
-     helpfile       help/nohelp.hlp
- }

WINDOW skkeyrdssearch {
    num_rows        10
    num_columns      70
    line            0,0, 0,77
    - datum         noiseff,      1,1, "CALL dummy"
    - datum         qualif,       1,30 ~ nr 2/5 Added
    - textblock     abdesc,       5,23
    - datum         noistype,     2,1, "CALL dummy"
    - datum         expdesc,     3,1, "CALL dummy"
    textline        notimplement, 5,20
    - textline      scrolltxt,     15,23
    button          stsearch,     17, 1, "CALL dummy"
    button          cansearch,    17,45, "CALL qsetup",
                                "NEW_SCREEN doing"
}

SCREEN skkeyrsh {
    title           "NOISE AND SONIC-BOOM EFFECTS KEYWORD SEARCH"
    window          skkeyrdssearch, 2,1
    - window        majoraction3, 10,1
    border          YES
}

-----
- Declaration for LEGISLATIVE effects keyword category search screen
-----

-WINDOW lkeyrdssearch {
-     num_rows      10
-     num_columns    70
-     line          0,0, 0,77
-     datum         noiseff,      1,1, "CALL dummy"
-     datum         qualif,       1,30 ~ nr 2/5 Added
-     datum         noistype,     2,1, "CALL dummy"
-     datum         expdesc,     3,1, "CALL dummy"
-     textblock     abdesc,       5,23
-     textline      andy,         5,20
-     textline      scrolltxt,     15,23

```

```

- button      stsearch, 17,1
- button      cansearch, 17,45, "CALL qsetup",
-              "NEW_SCREEN dbinq"
-   }

-SCREEN lkeywdschscreen {
- title       "LEGISLATIVE EFFECTS SEARCH BY KEYWORD"
- window      lkeywdssearch, 2,1
- window      majoraction3, 18,1
- border      YES
-   }

```

CITATION DISPLAY SCREEN

```
TEXTLINE shwreviewtxt ("SHOW REVIEW")
```

```

BUTTON shwreview {
  num_rows 1
  num_columns 15
  textline shwreviewtxt, 0,2
  helpfile help/citation.hlp
}

```

```
TEXTLINE dbinqtxt ("DATABASE INQUIRY SCREEN")
```

```

BUTTON dbinqir {
  num_rows 1
  num_columns 25
  textline dbinqtxt, 0, 2
  helpfile help/citation.hlp
}

```

```
TEXTLINE altsaltxt ("Alternative selections you can now make:")
```

```
TEXTLINE rescopetxt ("RESCOPE SEARCH")
```

```

BUTTON rescope {
  num_rows 1
  num_columns 27
  textline rescopetxt, 0, 2
  helpfile help/rescope.hlp
}

```

```

TEXTBLOCK slpintf {
  filename blrplt/slpintf.bpl
  num_rows 16
  num_columns 76
  border YES
}

```

```

WINDOW shwxtcit {
  num_rows 17
  num_columns 78
  textblock slpintf, 1,1
}

```

```
TEXTLINE shwxtcitxt ("Show next citation")
```

```

BUTTON shwxtcit {
  num_rows 1
  num_columns 22
  textline shwxtcitxt, 0, 2
  helpfile help/sohelp.hlp
}

```

```
TEXTLINE shwcritrvtxt ("Show critical review (if any)")
```

```

BUTTON shwcritrv {
  num_rows 1
  num_columns 35
  textline shwcritrvtxt, 0, 2
  helpfile help/sohelp.hlp
}

```

```
TEXTLINE shwabetxt ("Show abstract (if any)")
```

```
BUTTON shwabet {
```

```

num_rows      1
num_columns   24
textline      shwbetxt, 0, 2
helpfile      help/achelp.hlp
}

TEXTLINE prnthscitxt ("Print this citation")

BUTTON prnthscit {
num_rows      1
num_columns   24
textline      prnthscitxt, 0, 2
helpfile      help/achelp.hlp
}

TEXTLINE prncitxt ("PRINT ALL CITATIONS")

BUTTON prncit {
num_rows      1
num_columns   27
textline      prncitxt, 0, 2
helpfile      help/achelp.hlp
}

WINDOW citdisposition {
num_rows      2
num_columns   78
button        shwstcit, 0, 1, "CALL shwstcit"
button        shwbet, 0, 40, "CALL deplabet"
button        shwsttrv, 1, 1, "CALL deplarit"
button        prnthscit, 1, 40, "CALL dummy"
}

```

```

VARIABLE shwdesc1 {
foundin "VARCHAR60 entdesc[0].arr"
type     STRING
format   4-60s
}
DATUM shwdesc1 {
num_rows      1
num_columns   65
variable      shwdesc1, 0, 1
pickable     NO
}

```

```

VARIABLE shwdesc2 {
foundin "VARCHAR60 entdesc[1].arr"
type     STRING
format   4-60s
}
DATUM shwdesc2 {
num_rows      1
num_columns   65
variable      shwdesc2, 0, 1
pickable     NO
}

```

```

VARIABLE shwdesc3 {
foundin "VARCHAR60 entdesc[2].arr"
type     STRING
format   4-60s
}
DATUM shwdesc3 {
num_rows      1
num_columns   65
variable      shwdesc3, 0, 1
pickable     NO
}

```

```

VARIABLE shwdesc4 {
foundin "VARCHAR60 entdesc[3].arr"
type     STRING
format   4-60s
}
DATUM shwdesc4 {
num_rows      1
num_columns   65
}

```

```

variable    shwdesc4, 0, 1
pickable    NO
}

```

```

VARIABLE suitable {
  foundia "VARCHAR suitable.arr"
  type    STRING
  format  %1s
}
DATUM suitable {
  num_rows    1
  num_columns  23
  variable    suitable, 0, 21
  leader      "Suitability rating:"
  pickable    NO
}

```

```

VARIABLE entrynum {
  foundia "VARCHAR enumb.arr"
  type    STRING
  format  %4-6s
}
DATUM entrynum {
  num_rows    1
  num_columns  30
  variable    entrynum, 0, 23
  leader      "ANAN Citation number:"
  pickable    NO
}

```

```

TEXTLINE shwtitxt ("Title:")

```

```

WINDOW nitdisplay {
  num_rows    17
  num_columns  78
  datum       entrynum,    0, 1
  datum       suitable,    0, 32
  datum       when,        0, 60
  datum       citaenum1,   2, 1
  datum       citaenum2,   3, 1
  datum       citaenum3,   4, 1
  datum       citaenum4,   5, 1
  datum       citaenum5,   6, 1
  datum       citaenum6,   2, 42
  datum       citaenum7,   3, 42
  datum       citaenum8,   4, 42
  datum       citaenum9,   5, 42
  datum       citaenum10,  6, 42
  textline    shwtitxt,    8, 2
  datum       shwdesc1,    9, 9
  datum       shwdesc2,    9, 9
  datum       shwdesc3,   10, 9
  datum       shwdesc4,   11, 9
  button      shwnxtait,   14, 1, "ADD WINDOW shwnxtait  4 1" **** ??????
  button      prncit,     16, 1, "CALL dummy"
  button      rescope,    16, 28, "CALL mclista",
  button      cancoch,    16, 60, "CALL mclista",
                                "CALL qsetup",
                                "NEW_SCREEN doing"
}

```

```

VARIABLE salcrit {
  type    STRING
  format  %4-15s
}

```

```

DATUM salcrit {
  num_rows    1
  num_columns  70
  variable    salcrit, 0, 22
  leader      "Selection Criterion: "
  pickable    NO
}

```

```

WINDOW citdisphd {
    num_rows      3
    num_columns    78
    datum         salcrit,    0, 1
    datum         curcrit,    1, 1
    datum         sumofcit,    1,32
    line          2, 0, 2, 77
}

SCREEN citdepl {
    title "DETAILED DISPLAY OF RETRIEVED CITATIONS"
    window citdisphd,    2,1
    window citdisplay,    3,1
    border YES
}

-----
~ ABSTRACT/CRITICAL REVIEW DISPLAY SCREEN
-----

TEXTLINE nulltext (" ")

WINDOW shwxtabs {
    num_rows      1
    num_columns    15
    textline      nulltext, 0,2
}

TEXTLINE shwcritrv ("Show another review (if any)")

BUTTON shwxtrev {
    num_rows      1
    num_columns    24
    textline      shwcritrv, 0, 2
    helpfile      help/nohelp.hlp
}

WINDOW shwxtrev {
    num_rows      1
    num_columns    31
    button        shwxtrev, 0, 1, "REMOVE_WINDOW",
                                "CALL duplcrtr"
}

TEXTLINE abedone ("Done viewing this text")

BUTTON abedone {
    num_rows      1
    num_columns    29
    textline      abedone, 0, 2
    helpfile      help/nohelp.hlp
}

TEXTBLOCK memotext {
    num_rows      12
    num_columns    74
    filename      txtblk/memotxt.txt
}

WINDOW abedisplay {
    num_rows      17
    num_columns    77
    textline      shwtitxt,    1, 2
    datum         shwdesc1,    1, 9
    textblock     memotext,    3, 2
    button        abedone,    16,40, "REMOVE_WINDOW",
                                "NEW_SCREEN citdepl",
                                "ADD_WINDOW citdispaction 19 1"
}

SCREEN showabs {
    title "DISPLAY ABSTRACTS AND CRITICAL REVIEWS"
    window citdisphd,    2,1
    window abedisplay,    3,1
    border YES
}

-END OF ASANCIT.SDF

```

A.2 Screen Description File for MTR-related Operations

```
-----  
-INCLUDE STATEMENT FOR THE ASAN typedef DEFINITIONS NEEDED TO  
-    COMPILe USUBS.C WITHOUT INCURRING THE HEATH OF THE COMPILER  
-----
```

```
INCLUDE ASANTYPE.H  
INCLUDE ASAN.H
```

```
-----  
- DECLARATIONS IN NON-LEXICAL OBJECT SEQUENCE .....  
-----
```

```
- Multiple Choice Space  
-----
```

```
VARIABLE maldat00 {  
    foundin "VARCHAR30 deplmult[0].arr"  
    type    STRING  
    format  4-30s  
}
```

```
DATUM maldat00 {  
    num_rows    1  
    num_columns 32  
    variable    maldat00, 0, 2  
    pickable    NO  
}
```

```
VARIABLE maldat01 {  
    foundin "VARCHAR30 deplmult[1].arr"  
    type    STRING  
    format  4-30s  
}
```

```
DATUM maldat01 {  
    num_rows    1  
    num_columns 32  
    variable    maldat01, 0, 2  
    pickable    NO  
}
```

```
VARIABLE maldat02 {  
    foundin "VARCHAR30 deplmult[2].arr"  
    type    STRING  
    format  4-30s  
}
```

```
DATUM maldat02 {  
    num_rows    2  
    num_columns 32  
    variable    maldat02, 0, 2  
    pickable    NO  
}
```

```
VARIABLE maldat03 {  
    foundin "VARCHAR30 deplmult[3].arr"  
    type    STRING  
    format  4-30s  
}
```

```
DATUM maldat03 {  
    num_rows    1  
    num_columns 32  
    variable    maldat03, 0, 2  
    pickable    NO  
}
```

```
VARIABLE maldat04 {  
    foundin "VARCHAR30 deplmult[4].arr"  
    type    STRING  
    format  4-30s  
}
```

```
DATUM maldat04 {  
    num_rows    1  
    num_columns 32  
}
```



```

        variable    maldat04, 0, 2
        pickable    NO
    }

    VARIABLE maldat05 {
        foundia "VARCHAR30 deplmalt[5].arr"
        type    STRING
        format  4-30s
    }

    DATUM maldat05 {
        num_rows    1
        num_columns  32
        variable    maldat05, 0, 2
        pickable    NO
    }

    VARIABLE maldat06 {
        foundia "VARCHAR30 deplmalt[6].arr"
        type    STRING
        format  4-30s
    }

    DATUM maldat06 {
        num_rows    1
        num_columns  32
        variable    maldat06, 0, 2
        pickable    NO
    }

    VARIABLE maldat07 {
        foundia "VARCHAR30 deplmalt[7].arr"
        type    STRING
        format  4-30s
    }

    DATUM maldat07 {
        num_rows    1
        num_columns  32
        variable    maldat07, 0, 2
        pickable    NO
    }

    VARIABLE maldat08 {
        foundia "VARCHAR30 deplmalt[8].arr"
        type    STRING
        format  4-30s
    }

    DATUM maldat08 {
        num_rows    1
        num_columns  32
        variable    maldat08, 0, 2
        pickable    NO
    }

    VARIABLE maldat09 {
        foundia "VARCHAR30 deplmalt[9].arr"
        type    STRING
        format  4-30s
    }

    DATUM maldat09 {
        num_rows    1
        num_columns  32
        variable    maldat09, 0, 2
        pickable    NO
    }

    VARIABLE maldat10 {
        foundia "VARCHAR30 deplmalt[10].arr"
        type    STRING
        format  4-30s
    }

    DATUM maldat10 {
        num_rows    1
        num_columns  32
        variable    maldat10, 0, 2
        pickable    NO
    }

```

```

VARIABLE maldat11 {
  foundin "VARCHAR30 deplmult[11].arr"
  type    STRING
  format  4-30s
}

DATUM maldat11 {
  num_rows 1
  num_columns 32
  variable  maldat11, 0, 2
  pickable  NO
}

VARIABLE maldat12 {
  foundin "VARCHAR30 deplmult[12].arr"
  type    STRING
  format  4-30s
}

DATUM maldat12 {
  num_rows 1
  num_columns 32
  variable  maldat12, 0, 2
  pickable  NO
}

VARIABLE maldat13 {
  foundin "VARCHAR30 deplmult[13].arr"
  type    STRING
  format  4-30s
}

DATUM maldat13 {
  num_rows 1
  num_columns 32
  variable  maldat13, 0, 2
  pickable  NO
}

VARIABLE maldat14 {
  foundin "VARCHAR30 deplmult[14].arr"
  type    STRING
  format  4-30s
}

DATUM maldat14 {
  num_rows 1
  num_columns 32
  variable  maldat14, 0, 2
  pickable  NO
}

VARIABLE maldat15 {
  foundin "VARCHAR30 deplmult[15].arr"
  type    STRING
  format  4-30s
}

DATUM maldat15 {
  num_rows 1
  num_columns 32
  variable  maldat15, 0, 2
  pickable  NO
}

VARIABLE maldat16 {
  foundin "VARCHAR30 deplmult[16].arr"
  type    STRING
  format  4-30s
}

DATUM maldat16 {
  num_rows 1
  num_columns 32
  variable  maldat16, 0, 2
  pickable  NO
}

VARIABLE maldat17 {
  foundin "VARCHAR30 deplmult[17].arr"
  type    STRING
  format  4-30s
}

```

```

    }

    DATUM maldat17 {
        num_rows 1
        num_columns 32
        variable maldat17, 0, 2
        pickable NO
    }

    VARIABLE maldat18 {
        foundin "VARCHAR30 deplmalt[18].arr"
        type STRING
        format 4-30s
    }

    DATUM maldat18 {
        num_rows 1
        num_columns 32
        variable maldat18, 0, 2
        pickable NO
    }

    VARIABLE maldat19 {
        foundin "VARCHAR30 deplmalt[19].arr"
        type STRING
        format 4-30s
    }

    DATUM maldat19 {
        num_rows 1
        num_columns 32
        variable maldat19, 0, 2
        pickable NO
    }

    - Planner attributes
    -----

    VARIABLE planname {
        foundin "VARCHAR planname.arr"
        type STRING
        format 4-30s
    }

    DATUM planname {
        num_rows 1
        num_columns 50
        variable planname, 0, 20
        loader "Your name, please:"
        helpfile help/username.hlp
    }

    VARIABLE password {
        type STRING
        format 4-10s
    }

    DATUM password {
        num_rows 1
        num_columns 55
        variable password, 0, 23
        loader "Please enter password:"
        helpfile help/password.hlp
    }

    - Other General Storage for Verification, Tests, Etc.
    -----

    VARIABLE newstream {
        foundin "VARCHAR n2bv.arr"
        type STRING
        format 4-30s
    }

    - Noise Source Attributes
    -----

    VARIABLE srcid {
        foundin "VARCHAR srcid.arr"
        type STRING
        format 4-9s
    }

```

```

DATUM strans {
  num_rows 1
  num_columns 60
  variable sroid, 0, 21
  leader "Name of current MTR:"
  pickable NO
}

VARIABLE srodesc {
  foundin "VARCHAR srodesc.arr"
  type STRING
  format 4-54s
}

DATUM srods {
  num_rows 1
  num_columns 76
  variable srodesc, 0, 22
  leader "Description:"
  helpfile help/srtdesc.hlp
}

DATUM srtdesc {
  num_rows 1
  num_columns 70
  variable srodesc, 0, 7
  leader "Note: "
  pickable NO
}

VARIABLE datspabl {
  foundin "VARCHAR srodata.arr"
  type STRING
  format 4-12s
}

VARIABLE schedule {
  foundin "VARCHAR sroched.arr"
  type STRING
  format 4-50s
}

DATUM schedule {
  num_rows 1
  num_columns 73
  variable schedule, 0, 22
  leader "Scheduling activity:"
  helpfile help/schdctr.hlp
}

VARIABLE sroorig {
  foundin "VARCHAR sroorig.arr"
  type STRING
  format 4-50s
}

DATUM origstr {
  num_rows 1
  num_columns 73
  variable sroorig, 0, 22
  leader "Originating activity:"
  helpfile help/origstr.hlp
}

~ MTR Attributes
~ -----

VARIABLE curartoc {
  foundin "VARCHAR curartoc.arr"
  type STRING
  format 43s
}

~ CURRENT Set

DATUM curartoc {
  num_rows 1
  num_columns 27
  variable curartoc, 0, 16
  leader "ARTOC:"
  helpfile help/curartoc.hlp
}

VARIABLE curwidright {

```

```

    type    INTEGER
    format  %2d
}

DATUM curwidright {
    num_rows    1
    num_columns  25
    variable     curwidright, 0,16
    leader       "Width (right): "
    helpfile     help/curwidrt.hlp
}

VARIABLE curwidleft {
    type    INTEGER
    format  %2d
}

DATUM curwidleft {
    num_rows    1
    num_columns  25
    variable     curwidleft, 0,16
    leader       "Width (left): "
    helpfile     help/curwidlf.hlp
}

VARIABLE curhighalt {
    foundin  "ALTSPEC curhighalt.spec"
    type     STRING
    format   %3s
}

DATUM curhighalt {
    num_rows    1
    num_columns  27
    variable     curhighalt, 0,16
    leader       "High altitude: "
    helpfile     help/curhialt.hlp
}

VARIABLE curlowalt {
    foundin  "ALTSPEC curlowalt.spec"
    type     STRING
    format   %3s
}

VARIABLE preartec {
    foundin  "VARCHAR preartec.arr"
    type     STRING
    format   %3s
}

DATUM preartec {
    num_rows    1
    num_columns  8
    variable     preartec, 0,4
    pickable     NO
}

VARIABLE prewidright {
    type    INTEGER
    format  %2d
}

DATUM prewidright {
    num_rows    1
    num_columns  8
    variable     prewidright, 0,4
    pickable     NO
}

VARIABLE prewidleft {
    type    INTEGER
    format  %2d
}

DATUM prewidleft {
    num_rows    1
    num_columns  8
    variable     prewidleft, 0,4
    pickable     NO
}

```

- PREVIOUS Set

```

VARIABLE prehight {
  foundin "ALTSPEC prehight.spec"
  type STRING
  format %s
}

DATUM prehight {
  num_rows 1
  num_columns 10
  variable prehight, 0,1
  pickable NO
}

VARIABLE prelowlalt {
  foundin "ALTSPEC prelowlalt.spec"
  type STRING
  format %s
}

- Coordinates
- -----

VARIABLE entlist {
  foundin "COORDINATE ent.lst"
  type STRING
  format %-13s
}
- CURRENT or ENTER Set

VARIABLE entloag {
  foundin "COORDINATE ent.loc"
  type STRING
  format %-13s
}

VARIABLE shwlist {
  foundin "COORDINATE show.lst"
  type STRING
  format %-13s
}
- PREVIOUS or SHOW Set

VARIABLE shwloag {
  foundin "COORDINATE show.loc"
  type STRING
  format %-13s
}

- Navigation Point Properties
- -----

VARIABLE curfstyp {
  foundin "VARCHAR curfstyp.arr"
  type STRING
  format %12s
}
- CURRENT or ENTER Set

VARIABLE curfindist {
  type INTEGER
  format %3d
}

VARIABLE curfixrad {
  type INTEGER
  format %3d
}

VARIABLE curfixid {
  foundin "VARCHAR curfixid.arr"
  type STRING
  format %5s
}

VARIABLE curnavpt {
  foundin "VARCHAR curnavpt.arr"
  type STRING
  format %3s
}

VARIABLE prefistyp {
  foundin "VARCHAR prefistyp.arr"
  type STRING
  format %12s
}
- PREVIOUS or DISPLAY Set

```

```

VARIABLE prefindist {
    type      INTEGER
    format    %4d
}

VARIABLE prefixed {
    type      INTEGER
    format    %4d
}

VARIABLE prefmid {
    foundia   "VARCHAR prefmid.arr"
    type      STRING
    format    %4s
}

VARIABLE premsvpt {
    foundia   "VARCHAR premsvpt.arr"
    type      STRING
    format    %3s
}

- Aircraft Parameters
- -----

VARIABLE acname {
    foundia   "VARCHAR ac_name.arr"
    type      STRING
    format    %12s
}

DATUM acname {
    num_rows  1
    num_columns 25
    variable   acname, 0, 10
    leader     "Aircraft:"
    pickable   NO
}

VARIABLE acname1 {
    foundia   "VARCHAR tid.arr"
    type      STRING
    format    %12s
}

DATUM acname1 {
    num_rows  1
    num_columns 35
    variable   acname1, 0, 16
    leader     "Aircraft name:"
    helpfile   help/airairor.hlp
}

- Operations Specification
- -----

VARIABLE day {
    foundia   "OPERATIONS ops[0].day"
    type      INTEGER
    format    %4d
}

- GENERIC OF JANUARY

VARIABLE jday {
    foundia   "OPERATIONS ops[0].day"
    type      INTEGER
    format    %4d
}

VARIABLE night {
    foundia   "OPERATIONS ops[0].nite"
    type      INTEGER
    format    %4d
}

VARIABLE jannite {
    foundia   "OPERATIONS ops[0].nite"
    type      INTEGER
    format    %4d
}

VARIABLE febday {
    foundia   "OPERATIONS ops[1].day"

```

```

        type      INTEGER
        format     %4d
    )

    VARIABLE febsaite {
        foundin    "OPERATIONS ops[1].nite"
        type       INTEGER
        format     %4d
    }

    VARIABLE marday {
        foundin    "OPERATIONS ops[2].day"
        type       INTEGER
        format     %4d
    }

    VARIABLE maraite {
        foundin    "OPERATIONS ops[2].nite"
        type       INTEGER
        format     %4d
    }

    VARIABLE apnday {
        foundin    "OPERATIONS ops[3].day"
        type       INTEGER
        format     %4d
    }

    VARIABLE apraite {
        foundin    "OPERATIONS ops[3].nite"
        type       INTEGER
        format     %4d
    }

    VARIABLE mayday {
        foundin    "OPERATIONS ops[4].day"
        type       INTEGER
        format     %4d
    }

    VARIABLE mayaite {
        foundin    "OPERATIONS ops[4].nite"
        type       INTEGER
        format     %4d
    }

    VARIABLE junday {
        foundin    "OPERATIONS ops[5].day"
        type       INTEGER
        format     %4d
    }

    VARIABLE juanaite {
        foundin    "OPERATIONS ops[5].nite"
        type       INTEGER
        format     %4d
    }

    VARIABLE julday {
        foundin    "OPERATIONS ops[6].day"
        type       INTEGER
        format     %4d
    }

    VARIABLE julaite {
        foundin    "OPERATIONS ops[6].nite"
        type       INTEGER
        format     %4d
    }

    VARIABLE augday {
        foundin    "OPERATIONS ops[7].day"
        type       INTEGER
        format     %4d
    }

    VARIABLE augaite {
        foundin    "OPERATIONS ops[7].nite"
        type       INTEGER
        format     %4d
    }

```



```

VARIABLE sepday {
    foundin "OPERATIONS ops[8].day"
    type INTEGER
    format %4d
}

VARIABLE sepaite {
    foundin "OPERATIONS ops[8].nite"
    type INTEGER
    format %4d
}

VARIABLE octday {
    foundin "OPERATIONS ops[9].day"
    type INTEGER
    format %4d
}

VARIABLE octaite {
    foundin "OPERATIONS ops[9].nite"
    type INTEGER
    format %4d
}

VARIABLE novday {
    foundin "OPERATIONS ops[10].day"
    type INTEGER
    format %4d
}

VARIABLE novaita {
    foundin "OPERATIONS ops[10].nite"
    type INTEGER
    format %4d
}

VARIABLE decday {
    foundin "OPERATIONS ops[11].day"
    type INTEGER
    format %4d
}

VARIABLE decnita {
    foundin "OPERATIONS ops[11].nite"
    type INTEGER
    format %4d
}

~ Missions
~ -----

VARIABLE misname {
    foundin "VARCHAR misslabl.arr"
    type STRING
    format %7s
}

DATUM misname1 {
    num_rows 1
    num_columns 34
    variable misname, 0, 26
    leader "Name of current mission: "
    pickable NO
}

VARIABLE newmisname {
    foundin "VARCHAR cid.arr"
    type STRING
    format %7s
}

DATUM misname {
    num_rows 1
    num_columns 24
    variable newmisname, 0, 14
    leader "Mission name:"
    helpfile help/misname.hlp
}

TEXTLINE select {
    "Right now you can type ? for help, <CTRL> C to quit, or move the cursor"
}

TEXTLINE canvas {"CONDUCT AN ENVIRONMENTAL ASSESSMENT" }

```

```

TEXTLINE housekp ("PERFORM DATABASE HOUSEKEEPING" )
TEXTLINE introtxt ("VIEW GENERAL INFORMATION ABOUT THIS PROGRAM")

TEXTLINE selacmistr ("Select aircraft and mission for MFR")
TEXTLINE modcumstrtxt ("Modify current MFR")
TEXTLINE chgmstrtxt ("Select another MFR")

TEXTLINE nextnavptxt ("Enter next navigation point")
TEXTLINE curpretxt
{ "CURRENT      PREVIOUS          CURRENT      PREVIOUS" }
TEXTLINE curpretxt3
{ "-----" }
TEXTLINE cancelstrtxt ("Cancel this MFR data entry")
TEXTLINE savestrtxt ("Save this MFR")

TEXTLINE showmoremstrtxt ("Show more MFR names (if any)")
TEXTLINE recallmstrtxt ("Recall one of the following MFRs:")
TEXTLINE stnewmstrtxt ("Start new MFR ")
TEXTLINE definamstrtxt ("Enter route waypoints")

TEXTLINE aotxt ("Aircraft:")

TEXTLINE opseascontxt ("Operations are seasonal")
TEXTLINE opthruyrtxt ("Operations are even throughout year")
TEXTLINE instritxt ("Please enter day and night operations by month")
TEXTLINE daynite (" DAY NIGHT DAY NIGHT DAY NIGHT DAY NIGHT")
TEXTLINE instritxt ("Please enter daytime and night operations per month")
TEXTLINE cancmistrtxt ("Abandon this mission")
TEXTLINE entmisdtx ("Enter this mission into database")
TEXTLINE stnewmistrtxt ("Start new mission")

TEXTBLOCK introtxt {
    num_rows      14
    num_columns    76
    border        YES
    filename       txtblk/intro.txt
}

BUTTON assessment {
    num_rows      1
    num_columns    36
    textline      canvass, 0, 2
    helpfile      help/assess.hlp
}

BUTTON chgmtr {
    num_rows      1
    num_columns    35
    textline      chgmstrtxt, 0, 2
    helpfile      help/salmtr.hlp
}

BUTTON housekeeping {
    num_rows      1
    num_columns    33
    textline      housekp, 0, 2
    helpfile      help/housekp.hlp
}

BUTTON introtxt {
    num_rows      1
    num_columns    56
    textline      introtxt, 0, 2
    helpfile      help/introhel.hlp
}

BUTTON modcumtr {
    num_rows      1
    num_columns    35
    textline      modcumstrtxt, 0, 2
    helpfile      help/mohelp.hlp
    - no help here cuz this button doesn't do anything
}

WINDOW password {
    num_rows      1
    num_columns    60
    datum         password, 0, 5
}

TEXTLINE tititxt
("Developed for Noise and Sonic Boom Impact Technology Program")

```

```

TEXTLINE tit2txt ("under U.S. Air Force Contract F33615-86-C-0830")

TEXTLINE tit3txt ("by ESW Laboratories, Inc.")

TEXTLINE tit4txt ("February, 1988")

TEXTLINE tit5txt
("Unreleased demonstration of Prototype Version...Not for General Use")

TEXTLINE diafo ("Done viewing general information on ASAM")

BUTTON diafo {
    num_rows    1
    num_columns  76
    textline    diafo, 0, 2
}

WINDOW introduction {
    num_rows    15
    num_columns  78
    textblock   introtxt, 0, 1
    button      diafo, 14, 1, "REMOVE_WINDOW"
}

WINDOW introwindow1 {
    num_rows    21
    num_columns  78
    line        1, 0, 1, 77
    title       "ASSESSMENT SYSTEM FOR AIRCRAFT NOISE (ASAM)"
    textline    tit1txt, 2, 3
    textline    tit2txt, 3, 16
    textline    tit3txt, 4, 27
    textline    tit4txt, 5, 32
    textline    tit5txt, 7, 6
    line        8, 0, 8, 77
    datum       placnam, 12, 1,
                "CALL VCAPITAL $placnam",
                "ADD_WINDOW password 15 3",
                "UPDATE_DATUM password",
                "REMOVE_WINDOW",
                "CALL pscheck"
}

WINDOW introwindow2 {
    num_rows    9
    num_columns  78
    line        5, 0, 5, 77
    textline    select, 6, 3
    button      introtxt, 7, 15, "ADD_WINDOW introduction 3 1"
    button      assessment, 8, 2, "CALL psprobet"
    button      housekeeping, 9, 44, "CALL psdbasek"
}

SCREEN firstscreen {
    titlescreen YES
    border      YES
    title       "ASSESSMENT SYSTEM FOR AIRCRAFT NOISE (ASAM)"
    window      introwindow1, 1, 1
    window      introwindow2, 13, 1
}

BUTTON opseacon {
    num_rows    3
    num_columns  41
    textline    opseacontxt, 1, 3
    border      YES
    helpfile    help/vrblopc.hlp
}

BUTTON opteady {
    num_rows    3
    num_columns  41
    textline    opthruyrtxt, 1, 3
    border      YES
    helpfile    help/evnscpc.hlp
}

BUTTON mulbut00 {
    num_rows    1
    num_columns  1
    helpfile    help/mulbut.hlp
}

```

```

BUTTON mabut01 {
    num_rows 1
    num_columns 1
    helpfile help/mabut.hlp
}

BUTTON mabut02 {
    num_rows 1
    num_columns 1
    helpfile help/mabut.hlp
}

BUTTON mabut03 {
    num_rows 1
    num_columns 1
    helpfile help/mabut.hlp
}

BUTTON mabut04 {
    num_rows 1
    num_columns 1
    helpfile help/mabut.hlp
}

BUTTON mabut05 {
    num_rows 1
    num_columns 1
    helpfile help/mabut.hlp
}

BUTTON mabut06 {
    num_rows 1
    num_columns 1
    helpfile help/mabut.hlp
}

BUTTON mabut07 {
    num_rows 1
    num_columns 1
    helpfile help/mabut.hlp
}

BUTTON mabut08 {
    num_rows 1
    num_columns 1
    helpfile help/mabut.hlp
}

BUTTON mabut09 {
    num_rows 1
    num_columns 1
    helpfile help/mabut.hlp
}

BUTTON mabut10 {
    num_rows 1
    num_columns 2
    helpfile help/mabut.hlp
}

BUTTON mabut11 {
    num_rows 1
    num_columns 2
    helpfile help/mabut.hlp
}

BUTTON mabut12 {
    num_rows 1
    num_columns 2
    helpfile help/mabut.hlp
}

BUTTON mabut13 {
    num_rows 1
    num_columns 2
    helpfile help/mabut.hlp
}

BUTTON mabut14 {
    num_rows 1
    num_columns 2
    helpfile help/mabut.hlp
}

```

```

    }

    BUTTON mabut15 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut16 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut17 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut18 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut19 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

-----
-      Declarations for MTR DATA ENTRY SCREEN
-----

    BUTTON staemis {
        num_rows 1
        num_columns 40
        textline staemistxt, 0, 2
        helpfile help/staemis.hlp
    }

    BUTTON selacmis {
        num_rows 1
        num_columns 40
        textline selacmistxt, 0, 2
        helpfile help/selacm.hlp
    }

    TEXTLINE specwmistxt {"Specify new mission"}

    BUTTON specwmis {
        num_rows 1
        num_columns 30
        textline specwmistxt, 0, 2
        helpfile help/newmiss.hlp
    }

    WINDOW mtrdataentry {
        num_rows 8
        num_columns 78
        button okmtr, 0, 2, "CALL peachmtr"
        button selacmis, 2, 2, "CALL peachmis"
        button modcmtr, 4, 2, "CALL dummy"
        button specwmis, 6, 2, "ADD WINDOW newmiss 13 3",
            "UPDATE DATUM misname",
            "CALL WHEREAMI Screen Window Datum Button",
            "CALL stropy oldscreen Screen",
            "CALL peachmis aid.arr"
    }

    WINDOW mtrasmcom {
        num_rows 3
        num_columns 78
        line 2, 0, 2, 77
        datum mtrasm, 0, 2
        datum mtrdesc, 1, 2
    }

    WINDOW mtrasmocal {

```

```

num_rows      4
num_columns   78
line          3,0, 3,77
datum         stream, 0, 2
datum         stream, 1, 2
datum         stream, 2, 2
)

```

```

--      Declarations for DEFINE/MODIFY SCREENS (defmodstrscreen)

```

```

DATUM  prelowalt {
    num_rows      1
    num_columns   10
    variable      prelowalt, 0,1
    pickable      NO
}

DATUM  curlowalt {
    num_rows      1
    num_columns   29
    variable      curlowalt, 0, 16
    leader        "Low altitude: "
    helpfile      help/curlowalt.hlp
}

DATUM  prefirtype {
    num_rows      1
    num_columns   13
    variable      prefirtype, 0,0
    pickable      NO
}

DATUM  curfirtype {
    num_rows      1
    num_columns   24
    variable      curfirtype, 0,11
    leader        "Fir type: "
    helpfile      help/curfirtype.hlp
}

```

```

--      Declarations for entering coordinates

```

```

DATUM  entlong {
    num_rows      1
    num_columns   25
    variable      entlong, 0, 11
    leader        "Longitude: "
    helpfile      help/markmap.hlp
}

DATUM  entlat {
    num_rows      1
    num_columns   25
    variable      entlat, 0, 11
    leader        "Latitude: "
    helpfile      help/markmap.hlp
}

```

```

--      Declarations for show coordinates

```

```

DATUM  shwlat {
    num_rows      1
    num_columns   14
    variable      shwlat, 0, 0
    pickable      NO
    helpfile      help/combmap.hlp
}

DATUM  shwlong {
    num_rows      1
    num_columns   14
    variable      shwlong, 0, 0
    pickable      NO
    helpfile      help/combmap.hlp
}

```

```

DATUM prefindist {
    num_rows      1
    num_columns    8
    variable       prefindist, 0,4
    pickable       NO
}

DATUM curfindist {
    num_rows      1
    num_columns    26
    variable       curfindist, 0,16
    leader         "Fix distance: "
    helpfile       help/curfindi.hlp
}

DATUM prefixrad {
    num_rows      1
    num_columns    8
    variable       prefixrad, 0,4
    pickable       NO
}

DATUM curfixrad {
    num_rows      1
    num_columns    26
    variable       curfixrad, 0,16
    leader         "Fix radial: "
    helpfile       help/curfixra.hlp
}

DATUM prefixid {
    num_rows      1
    num_columns    10
    variable       prefixid, 0,4
    pickable       NO
}

DATUM curfixid {
    num_rows      1
    num_columns    26
    variable       curfixid, 0,16
    leader         "Fix ID: "
    helpfile       help/curfixid.hlp
}

DATUM prenavpt {
    num_rows      1
    num_columns    10
    variable       prenavpt, 0,4
    pickable       NO
}

DATUM curnavpt {
    num_rows      1
    num_columns    24
    variable       curnavpt, 0, 16
    leader         "Nav. Point: "
    helpfile       help/curnavpt.hlp
}

DATUM navstrm {
    num_rows      1
    num_columns    40
    variable       navstrm, 0,7
    leader         "Name:"
    helpfile       help/ntmstrm.hlp
}

BUTTON nextnavpt {
    num_rows      1
    num_columns    35
    textline       nextnavptxt, 0, 2
    helpfile       help/ntnav.hlp
}

TEXTLINE retntreatxt ("Continue without selecting MFR")
TEXTLINE retntreatxt1 ("Continue without creating mission")
TEXTLINE retntreatxt2 ("Save mission in database")
TEXTLINE retnomis ("Continue without selecting mission")

```

```

BUTTON retstrent {
    num_rows      1
    num_columns   40
    textline      retstrentxt, 0, 2
    helpfile      help/noneentr.hlp
}

```

```

BUTTON retstrent1 {
    num_rows      1
    num_columns   40
    textline      retstrentxt1, 0, 2
    helpfile      help/noneiss.hlp
}

```

```

BUTTON retstrent2 {
    num_rows      1
    num_columns   30
    textline      retstrentxt2, 0, 2
    helpfile      help/savemiss.hlp
}

```

```

BUTTON retstomis {
    num_rows      1
    num_columns   40
    textline      retstomis, 0, 2
    helpfile      help/noneemis.hlp
}

```

```

WINDOW defmodetr {
    num_rows      15
    num_columns   78
    textline      curpretxt, 0, 16
    textline      curpretxt3, 1, 16
    datum        curasvpt, 2, 1, "CALL VCAPITAL &curasvpt",
    "NEWVALS"
    datum        preasvpt, 2, 26
    datum        curfixid, 3, 1, "CALL VCAPITAL &curfixid",
    "NEWVALS"
    datum        prefixid, 3, 26
    datum        curfixrad, 4, 1
    datum        prefixrad, 4, 26
    datum        curfixdist, 5, 1
    datum        prefixdist, 5, 26
    datum        entlat, 2, 38, "CALL lat2dec &ent",
    "NEWVALS"
    datum        shwlat, 2, 64
    datum        entlong, 3, 38, "CALL lon2dec &ent",
    "NEWVALS"
    datum        shwlong, 3, 64
    datum        curfixtype, 4, 38, "CALL VCAPITAL &curfixtyp",
    "NEWVALS"
    datum        prefixtype, 4, 64
    datum        curlowalt, 7, 2, "CALL alt2dec &curlowalt",
    "NEWVALS"
    datum        prelowalt, 7, 26
    datum        curhighalt, 8, 2, "CALL alt2dec &curhighalt",
    "NEWVALS"
    datum        prehighalt, 8, 26
    datum        curwidleft, 9, 2
    datum        prewidleft, 9, 27
    datum        curwidright, 10, 2
    datum        prewidright, 10, 27
    datum        curartec, 11, 2, "CALL VCAPITAL &curartec",
    "NEWVALS"
    datum        preartec, 11, 26
    button       nextasvpt, 13, 2, "CALL nexttrpt"
}

```

```

BUTTON cancelmtr {
    num_rows      1
    num_columns   35
    textline      cancelmtrtxt, 0, 2
    helpfile      help/cancelmtr.hlp
}

```

```

BUTTON savemtr {
    num_rows      1
    num_columns   20
    textline      savemtrtxt, 0, 2
    helpfile      help/savemtr.hlp
}

```



```

    }

WINDOW ntraction {
    num_rows      2
    num_columns    78
    line          0,0, 0,77
    button        savestr,  1, 2, "CALL savestr"
    button        cancelstr, 1,40, "CALL cancelstr"
}

```

```

SCREEN defmodatrscreen {
    title         "DEFINE/MODIFY MFR"
    window        mtrnamecom, 2, 1
    window        defmodstr, 6, 1
    window        ntraction, 20, 1
    border        YES
}

```

 " Declarations for SELECT ANOTHER MFR SCREEN

```

BUTTON showmorestr {
    num_rows      1
    num_columns    37
    textline      showmorestrtxt, 0,2
    helpfile      help/showmore.hlp
}

```

```

BUTTON stnamestr {
    num_rows      1
    num_columns    40
    textline      stnamestrtxt, 0,2
    helpfile      help/startstr.hlp
}

```

```

WINDOW newstrm {
    num_rows      1
    num_columns    40
    datum         newstrm, 0, 2
}

```

```

WINDOW strecalstr {
    num_rows      4
    num_columns    78
    button        stnamestr, 1, 2,
                  "ADD WINDOW newstrm 6 3",
                  "UPDATE DATUM newstrm",
                  "CALL penstrm a2bv.arr"
    button        showmorestr, 1, 41,
                  "CALL mabunch"
    textline      recallstrtxt, 3, 2
}

```

```

WINDOW curdat {
    num_rows      10
    num_columns    32
    datum         maldat00, 0, 0
    datum         maldat01, 1, 0
    datum         maldat02, 2, 0
    datum         maldat03, 3, 0
    datum         maldat04, 4, 0
    datum         maldat05, 5, 0
    datum         maldat06, 6, 0
    datum         maldat07, 7, 0
    datum         maldat08, 8, 0
    datum         maldat09, 9, 0
}

```

```

WINDOW curdat1 {
    num_rows      10
    num_columns    32
    datum         maldat10, 0, 0
    datum         maldat11, 1, 0
    datum         maldat12, 2, 0
    datum         maldat13, 3, 0
    datum         maldat14, 4, 0
    datum         maldat15, 5, 0
    datum         maldat16, 6, 0
    datum         maldat17, 7, 0
    datum         maldat18, 8, 0
    datum         maldat19, 9, 0
}

```

```

}

WINDOW countsbet {
    sum_rows 10
    sum_columns 2
    button m1bet00, 0, 0, "CALL MTRocan deplnult[0].arr"
    button m1bet01, 1, 0, "CALL MTRocan deplnult[1].arr"
    button m1bet02, 2, 0, "CALL MTRocan deplnult[2].arr"
    button m1bet03, 3, 0, "CALL MTRocan deplnult[3].arr"
    button m1bet04, 4, 0, "CALL MTRocan deplnult[4].arr"
    button m1bet05, 5, 0, "CALL MTRocan deplnult[5].arr"
    button m1bet06, 6, 0, "CALL MTRocan deplnult[6].arr"
    button m1bet07, 7, 0, "CALL MTRocan deplnult[7].arr"
    button m1bet08, 8, 0, "CALL MTRocan deplnult[8].arr"
    button m1bet09, 9, 0, "CALL MTRocan deplnult[9].arr"
}

WINDOW countsbet1 {
    sum_rows 12
    sum_columns 2
    button m1bet10, 0, 0, "CALL MTRocan deplnult[10].arr"
    button m1bet11, 1, 0, "CALL MTRocan deplnult[11].arr"
    button m1bet12, 2, 0, "CALL MTRocan deplnult[12].arr"
    button m1bet13, 3, 0, "CALL MTRocan deplnult[13].arr"
    button m1bet14, 4, 0, "CALL MTRocan deplnult[14].arr"
    button m1bet15, 5, 0, "CALL MTRocan deplnult[15].arr"
    button m1bet16, 6, 0, "CALL MTRocan deplnult[16].arr"
    button m1bet17, 7, 0, "CALL MTRocan deplnult[17].arr"
    button m1bet18, 8, 0, "CALL MTRocan deplnult[18].arr"
    button m1bet19, 9, 0, "CALL MTRocan deplnult[19].arr"
}

WINDOW retstrest {
    sum_rows 1
    sum_columns 45
    button retstrest, 0, 1, "CALL m1stc",
    "CALL pmstrest"
}

SCREEN objcountscreen {
    title "SELECT ANOTHER MTR"
    window strmcom, 2, 1
    window strcalmtr, 5, 1
    window countsbet, 9, 3
    window countsbet1, 9, 42
    window curdat, 9, 5
    window curdat1, 9, 44
    window retstrest, 20, 2
    border YES
}

-----
- Declarations for new mtr name
-----

DATUM datapubl {
    sum_rows 1
    sum_columns 50
    variable datapubl, 0, 22
    leader "Date of publication: "
    helpfile help/datapubl.hlp
}

BUTTON definamtr {
    sum_rows 1
    sum_columns 40
    textline definamtrtxt, 0, 2
    helpfile help/getmap.hlp
}

WINDOW defamtr {
    sum_rows 9
    sum_columns 77
    datum origmtr, 0, 1
    datum schedule, 2, 1
    datum erods, 4, 1
    datum datapubl, 6, 1
    button definamtr, 8, 1, "CALL entmtrpt"
}

SCREEN mtrdefinescreen {
    title "MTR DEFINITION"
}

```

```

window  mtrmemcom,      3,1
window  defawmtr,       7,2
window  mtraction,     20,1
border   YES
}

```

Declarations for MISSION REQUIREMENTS WINDOW

```

VARIABLE  as_in_form {
    type      INTEGER
    format     %2d
    lowlimit   1
    uplimit    16
    default    2
}

VARIABLE  perstunt {
    foundin    "VARCHAR pr_pwr_u.arr"
    type       STRING
    format     %s
}

DATUM     perstunt {
    num_rows   1
    num_columns 10
    variable    perstunt, 0, 2
    leader      "["
    trailer     "]"
    pickable    NO
}

VARIABLE  mistype {
    type       STRING
    format     %s
}

DATUM     mistype1 {
    num_rows   1
    num_columns 18
    variable    mistype, 0, 14
    leader      "Mission type:"
    helpfile    help/mistype.hlp
}

DATUM     mistype {
    num_rows   1
    num_columns 18
    variable    mistype, 0, 14
    leader      "Mission type:"
    pickable    NO
}

VARIABLE  prealtlev {
    foundin    "ALTSPFC prelowalt.spec"
    type       STRING
    format     %s
}

DATUM     prealtlev {
    num_rows   1
    num_columns 10
    variable    prealtlev, 0, 0
    pickable    NO
}

VARIABLE  curaltlev {
    foundin    "ALTSPFC curlowalt.spec"
    type       STRING
    format     %s
}

DATUM     curaltlev {
    num_rows   1
    num_columns 21
    variable    curaltlev, 0, 11
    leader      "Alt: "
    helpfile    help/altmtr.hlp
}

VARIABLE  as_pwr {

```

```

        type      DOUBLE
        format    %12.3lf
        lowlimit  0.0
        uplimit   4000.0
        default   100.0
    }

    DATUM  prepwrset {
        num_rows      1
        num_columns    12
        variable       ac_pre_pwr, 0, 0
        pickable       NO
    }

    VARIABLE ac_cur_pwr {
        type      DOUBLE
        format    %10.3lf
        lowlimit  0.0
        uplimit   4000.0
        default   100.0
    }

    DATUM  curwrset {
        num_rows      1
        num_columns    21
        variable       ac_cur_pwr, 0, 11
        leader         "Power: "
        helpfile       help/pwrwrset.hlp
    }

    VARIABLE ac_pre_spd {
        type      INTEGER
        format    %3d
    }

    DATUM  prespeed {
        num_rows      1
        num_columns    9
        variable       ac_pre_spd, 0, 0
        trailer        "KTS"
        pickable       NO
    }

    VARIABLE ac_cur_spd {
        type      INTEGER
        format    %3d
        lowlimit  100
        uplimit   600
        default   450
    }

    DATUM  curspeed {
        num_rows      1
        num_columns    21
        variable       ac_cur_spd, 0, 11
        leader         "Speed: "
        trailer        "KTS"
        helpfile       help/srpspeed.hlp
    }

    DATUM  presavpt1 {
        num_rows      1
        num_columns    3
        variable       presavpt, 0, 0
        pickable       NO
    }

    DATUM  cursavpt1 {
        num_rows      1
        num_columns    21
        variable       cursavpt, 0, 11
        leader         "Waypoint:"
        helpfile       help/savname.hlp
    }

    TEXTLINE cancelparatxt ("Cancel data entry for this mission")

    BUTTON  cancelpara {
        num_rows      1
        num_columns    35
        textline       cancelparatxt, 0, 2
        helpfile       help/cancelmtr.hlp
    }

```

```

    }

TEXTLINE saveparatxt ("Save this mission's data")

BOTTOM savepara {
    num_rows      1
    num_columns    30
    textline       saveparatxt, 0, 2
    helpfile       help/savemiss.hlp
}

TEXTLINE opardatatxt ("Ready to enter operational data")

BOTTOM opardata {
    num_rows      1
    num_columns    33
    textline       opardatatxt, 0, 2
    helpfile       help/entropc.hlp
}

TEXTBLOCK mtr {
    num_rows      10
    num_columns    38
    filename       tutblk/mtr.txt
}

TEXTLINE strlabeltxt {"NAV    FIX    FIX TYPE    WIDTH"}

TEXTLINE strtxt {"REFERENCE INFORMATION FOR DATA ENTRY"}

TEXTLINE curpretat1 {"CURRENT    PREVIOUS"}

TEXTLINE curpretat2 {"-----    -----"}

WINDOW opentry {
    num_rows      10
    num_columns    78
    button opsteady, 2, 19, "CALL psaddops 1"
    button opseason, 7, 19, "CALL psaddops 12"
}

WINDOW fltpara {
    num_rows      17
    num_columns    78
    datum         mistype, 0, 2
    datum         acname, 1, 2
    textline       strtxt, 0, 40
    textline       strlabeltxt, 2, 38
    textline       curpretat1, 3, 14
    textline       curpretat2, 4, 14
    datum         curnavpt1, 5, 1, "CALL vcapital &curnavpt",
                                "NEWVALS"
    datum         presnavpt1, 5, 25
    datum         curspeed, 7, 1
    datum         prespeed, 7, 25
    datum         curprset, 9, 2
    datum         prepreset, 9, 22
    datum         pwrstunt, 10, 15
    datum         curaltlev, 12, 2, "CALL alt2dec &curlowalt",
                                "NEWVALS"
    datum         prealtlev, 12, 23
    box            1, 37, 14, 77
    line           3, 38, 3, 76
    line           15, 0, 15, 77
    button         nextnavpt, 14, 2, "CALL nextnavpt"
    button         opardata, 16, 2, "NEW_SCREEN opentry"
    button         canclpara, 16, 40, "CALL canclmis"
    textblock      mtr, 4, 38
}

SCREEN mtrflt {
    title          "FLIGHT PARAMETER ENTRY"
    window         mtrnameom, 2, 1
    window         fltpara, 5, 1
    border         YES
}

SCREEN opentry {
    title          "FLIGHT OPERATION DATA ENTRY FOR MTR"
    window         mtrnameom, 3, 1
    window         opentry, 6, 1
    border         YES
}

```

}

- Declarations for seasonal MTR and REQUIREMENTS

```

DATUM  decnite {
    num_rows 1
    num_columns 6
    variable  decnite, 0, 0
    helpfile  help/mnthnrite.hlp
}

DATUM  decday {
    num_rows 1
    num_columns 10
    variable  decday, 0, 5
    leader    "DEC:"
    helpfile  help/mnthday.hlp
}

DATUM  novnite {
    num_rows 1
    num_columns 6
    variable  novnite, 0, 0
    helpfile  help/mnthnrite.hlp
}

DATUM  novday {
    num_rows 1
    num_columns 10
    variable  novday, 0, 5
    leader    "NOV:"
    helpfile  help/mnthday.hlp
}

DATUM  octnite {
    num_rows 1
    num_columns 6
    variable  octnite, 0, 0
    helpfile  help/mnthnrite.hlp
}

DATUM  octday {
    num_rows 1
    num_columns 10
    variable  octday, 0, 5
    leader    "OCT:"
    helpfile  help/mnthday.hlp
}

DATUM  sepnite {
    num_rows 1
    num_columns 6
    variable  sepnite, 0, 0
    helpfile  help/mnthnrite.hlp
}

DATUM  sepday {
    num_rows 1
    num_columns 10
    variable  sepday, 0, 5
    leader    "SEP:"
    helpfile  help/mnthday.hlp
}

DATUM  augnite {
    num_rows 1
    num_columns 6
    variable  augnite, 0, 0
    helpfile  help/mnthnrite.hlp
}

DATUM  augday {
    num_rows 1
    num_columns 10
    variable  augday, 0, 5
    leader    "AUG:"
    helpfile  help/mnthday.hlp
}

DATUM  julaite {
    num_rows 1

```

```

        num_columns 6
        variable julnite, 0, 0
        helpfile help/mathnite.hlp
    }

    DATUM julday {
        num_rows 1
        num_columns 10
        variable julday, 0, 5
        leader "JUL:"
        helpfile help/mathday.hlp
    }

    DATUM junnite {
        num_rows 1
        num_columns 6
        variable junnite, 0, 0
        helpfile help/mathnite.hlp
    }

    DATUM junday {
        num_rows 1
        num_columns 10
        variable junday, 0, 5
        leader "JUN:"
        helpfile help/mathday.hlp
    }

    DATUM maynite {
        num_rows 1
        num_columns 6
        variable maynite, 0, 0
        helpfile help/mathnite.hlp
    }

    DATUM mayday {
        num_rows 1
        num_columns 10
        variable mayday, 0, 5
        leader "MAY:"
        helpfile help/mathday.hlp
    }

    DATUM aprnite {
        num_rows 1
        num_columns 6
        variable aprnite, 0, 0
        helpfile help/mathnite.hlp
    }

    DATUM aprday {
        num_rows 1
        num_columns 10
        variable aprday, 0, 5
        leader "APR:"
        helpfile help/mathday.hlp
    }

    DATUM marnite {
        num_rows 1
        num_columns 6
        variable marnite, 0, 0
        helpfile help/mathnite.hlp
    }

    DATUM marday {
        num_rows 1
        num_columns 10
        variable marday, 0, 5
        leader "MAR:"
        helpfile help/mathday.hlp
    }

    DATUM febnite {
        num_rows 1
        num_columns 6
        variable febnite, 0, 0
        helpfile help/mathnite.hlp
    }

    DATUM febday {
        num_rows 1
        num_columns 10

```

```

        variable    febday, 0, 5
        leader      "FEB:"
        helpfile    help/mnthday.hlp
    }

    DATUM  janaita {
        num_rows 1
        num_columns 6
        variable  janaita, 0, 0
        helpfile  help/mnthaita.hlp
    }

    DATUM  janaday {
        num_rows 1
        num_columns 10
        variable  janaday, 0, 5
        leader    "JAN:"
        helpfile  help/mnthday.hlp
    }

    WINDOW  month {
        num_rows 14
        num_columns 77
        textline instrtxt, 1, 1
        textline dayaita, 3, 0
        datum  janaday, 4, 1
        datum  janaita, 4, 12
        datum  febday, 5, 1
        datum  febaite, 5, 12
        datum  marday, 6, 1
        datum  maraita, 6, 12
        datum  aprday, 4, 20
        datum  apraita, 4, 31
        datum  mayday, 5, 20
        datum  mayaita, 5, 31
        datum  junday, 6, 20
        datum  janaita, 6, 31
        datum  julday, 4, 39
        datum  julaita, 4, 50
        datum  augday, 5, 39
        datum  augaita, 5, 50
        datum  sepday, 6, 39
        datum  sepaite, 6, 50
        datum  octday, 4, 58
        datum  octaita, 4, 69
        datum  novday, 5, 58
        datum  novaite, 5, 69
        datum  decday, 6, 58
        datum  decaita, 6, 69
        button savepara, 13, 2, "CALL savepara"
        button cancelpara, 13, 40, "CALL cancelpara"
    }

```

- Declarations for new SELECT AIRCRAFT AND MISSION FOR MTR

```

TEXTLINE  shmoremistxt {"Show more mission names (if any)"}

BUTTON  shmoremis {
    num_rows 1
    num_columns 38
    textline shmoremistxt, 0, 2
    helpfile  help/shmmiss.hlp
}

TEXTLINE  recalmis {"Recall one of the following missions"}

WINDOW  newmis {
    num_rows 1
    num_columns 40
    datum  misname, 0, 2
}

WINDOW  garmis {
    num_rows 16
    num_columns 78
    button  stasemis, 1, 2, "ADD_WINDOW newmis 7 3",
        "UPDATE_DATUM misname",
        "CALL penmisaid.ari"
    button  shmoremis, 1, 40, "CALL sabunch"
    textline  recalmis, 3, 2
}

```



```

datum maldat00, 4, 4
datum maldat01, 5, 4
datum maldat02, 6, 4
datum maldat03, 7, 4
datum maldat04, 8, 4
datum maldat05, 9, 4
datum maldat06, 10, 4
datum maldat07, 11, 4
datum maldat08, 12, 4
datum maldat09, 13, 4
datum maldat10, 4, 42
datum maldat11, 5, 42
datum maldat12, 6, 42
datum maldat13, 7, 42
datum maldat14, 8, 42
datum maldat15, 9, 42
datum maldat16, 10, 42
datum maldat17, 11, 42
datum maldat18, 12, 42
datum maldat19, 13, 42
button mabut00, 4, 2, "CALL MISsion deplmalt[0].arr"
button mabut01, 5, 2, "CALL MISsion deplmalt[1].arr"
button mabut02, 6, 2, "CALL MISsion deplmalt[2].arr"
button mabut03, 7, 2, "CALL MISsion deplmalt[3].arr"
button mabut04, 8, 2, "CALL MISsion deplmalt[4].arr"
button mabut05, 9, 2, "CALL MISsion deplmalt[5].arr"
button mabut06, 10, 2, "CALL MISsion deplmalt[6].arr"
button mabut07, 11, 2, "CALL MISsion deplmalt[7].arr"
button mabut08, 12, 2, "CALL MISsion deplmalt[8].arr"
button mabut09, 13, 2, "CALL MISsion deplmalt[9].arr"
button mabut10, 4, 40, "CALL MISsion deplmalt[10].arr"
button mabut11, 5, 40, "CALL MISsion deplmalt[11].arr"
button mabut12, 6, 40, "CALL MISsion deplmalt[12].arr"
button mabut13, 7, 40, "CALL MISsion deplmalt[13].arr"
button mabut14, 8, 40, "CALL MISsion deplmalt[14].arr"
button mabut15, 9, 40, "CALL MISsion deplmalt[15].arr"
button mabut16, 10, 40, "CALL MISsion deplmalt[16].arr"
button mabut17, 11, 40, "CALL MISsion deplmalt[17].arr"
button mabut18, 12, 40, "CALL MISsion deplmalt[18].arr"
button mabut19, 13, 40, "CALL MISsion deplmalt[19].arr"
button retcomis, 15, 2, "CALL selistc",
                        "CALL NEW_SCREEN oldscreen"
)

TEXTLINE salmisstxt ("Select mission")

BUTTON salmiss {
    num_rows 1
    num_columns 38
    textline salmisstxt, 0, 2
    helpfile help/salmiss.hlp
}

TEXTLINE fltparinfxt ("Specify flight parameter information")

BUTTON fltparinf {
    num_rows 1
    num_columns 37
    textline fltparinfxt, 0, 2
    helpfile help/fltparam.hlp
}

WINDOW salacmiss {
    num_rows 15
    num_columns 77
    datum achannel, 1, 1, "CALL vfyacmtr tid.arr"
    button salmiss, 3, 2, "CALL pchgmis"
    button fltparinf, 8, 2, "CALL pmtrflt"
    button retstramt1, 13, 2, "CALL cancmis"
}

SCREEN spstrmis {
    title "SELECT AIRCRAFT AND MISSION FOR MFR"
    window strmacom1, 3, 1
    window salacmiss, 7, 1
    border YES
}

SCREEN chgmis {
    title "SELECT AIRCRAFT AND MISSION FOR MFR"
    window strmacom1, 2, 1
    window curmiss, 6, 1

```

```
border    YES
}
```

Declarations for Day-Night window

```
DATUM night {
  num_rows      1
  num_columns    16
  variable       night, 0, 8
  leader         "Night:"
  helpfile       help/nightope.hlp
}
```

```
DATUM day {
  num_rows      1
  num_columns    16
  variable       day, 0, 6
  leader         "Day:"
  helpfile       help/dayope.hlp
}
```

```
WINDOW daynite {
  num_rows      12
  num_columns    77
  datum         scene,      1, 1
  textlines instrtext,     3, 1
  datum         day,        5, 2
  datum         night,     8,15
  button savepara, 10, 2, "CALL ascope",
                        "CALL saveas"
  button cancelpara, 10,40, "CALL onamis"
  border        YES
}
```

Declarations for mission specification window

```
DATUM sortie {
  num_rows      1
  num_columns    37
  variable       ac_in form, 0, 35
  leader         "Number of aircraft in formation: "
  helpfile       help/numform.hlp
}
```

```
VARIABLE misdesc {
  foundin "VARIABLE misdesc.txt"    -this description is for mission
  type     STRING
  format   4-60s
}
```

```
DATUM misdesc1 {
  num_rows      1
  num_columns    76
  variable       misdesc, 0, 7
  leader         "Descr:"
  pickable       NO
}
```

```
DATUM misdesc {
  num_rows      1
  num_columns    76
  variable       misdesc, 0, 14
  leader         "Description:"
  helpfile       help/misdesc.hlp
}
```

```
WINDOW strnamoc1 {
  num_rows      3
  num_columns    78
  line          2,0, 2,77
  datum         misnamel, 0, 2
  datum         misdesc1, 1, 2
}
```

```
WINDOW mispec {
  num_rows      16
  num_columns    77
  datum         misdesc, 1,1
}
```

```

datum mistype1, 3,2
datum sortie, 3,2
button retntreat1, 14,2, "CALL NEW_SCREEN oldscreen"
button retntreat2, 14,47, "CALL antennis"
}

SCREEN misspec {
    title "MISSION SPECIFICATION FOR AN MTA"
    window xtrasmocm3, 3,1
    window misspec, 6,1
    border YES
}

-01-14-88 st will try to fix this file so it will be only for assessments
-01-14-88 st changes will be commented with date
-01-16-88 st change title on first screen, change int. long. format loc
" map control
-01-17-88 st can't to fix sdf file commented with 1-17-88

-----
~ DECLARATIONS FOR SCREEN HEADER
-----

VARIABLE assename {
    foundin "ASAMHEADR ASSESSMENT.name"
    type STRING
    format %-30s
}

DATUM assename {
    num_rows 1
    num_columns 30
    variable assename, 0, 18
    leader "Assessment name: "
    pickable NO
}

DATUM assename1 {
    num_rows 1
    num_columns 75
    variable assename, 0, 30
    leader "Name of current assessment: "
    pickable NO
}

VARIABLE comment {
    foundin "ASAMHEADR ASSESSMENT.desc"
    type STRING
    format %-66s
}

DATUM comment {
    num_rows 1
    num_columns 76
    variable comment, 0, 10
    leader "Comment: "
    pickable NO
}

VARIABLE currmapnam {
    type STRING
    format %-30s
}

DATUM currmapnam {
    num_rows 1
    num_columns 77
    variable currmapnam, 0, 19
    leader "Current map name: "
    pickable NO
}

WINDOW lin {
    num_rows 1
    num_columns 78
    line 0, 0, 0,77
}

WINDOW assenmocom {
    num_rows 3
    num_columns 78
    datum assename1, 0,1

```

```

        datum      comment, 1, 2
        line       2, 0, 2,77
    )

WINDOW assessmcol {
    num_rows      4
    num_columns    78
    datum         assessmcol, 0, 0
    datum         comment, 1, 1
    datum         curmapass, 2, 1
    line          3, 0, 3,77
}

-----
-   Declarations for buttons used in majoraction footer
-----

TEXTLINE probstat1 ( "REVIEW CURRENT ASSESSMENT STATUS" )

BUTTON probstat {
    num_rows      1
    num_columns   35
    textline      probstat1, 0, 2
    helpfile      help/probstat.hlp
}

TEXTLINE probstat2 ( "REVIEW ASSESSMENT STATUS" )

BUTTON probstat2 {
    num_rows      1
    num_columns   26
    textline      probstat2, 0, 2
    helpfile      help/revstat.hlp
}

TEXTLINE probdef ( "ADD INFORMATION TO CURRENT ASSESSMENT" )

BUTTON probdef {
    num_rows      1
    num_columns   39
    textline      probdef, 0, 2
    helpfile      help/probdef.hlp
}

TEXTLINE probdef1 ( "ADD TO ASSESSMENT DEFINITION" )

BUTTON probdef1 {
    num_rows      1
    num_columns   31
    textline      probdef1, 0, 2
    helpfile      help/probdef.hlp
}

TEXTLINE analysis ( "ANALYSE DATA" )

BUTTON analysis {
    num_rows      1
    num_columns   15
    textline      analysis, 0, 2
    helpfile      help/datanal.hlp
}

TEXTLINE reportgen ( "MAKE A REPORT" )

BUTTON reportgen {
    num_rows      1
    num_columns   16
    textline      reportgen, 0, 2
    helpfile      help/reportgen.hlp
}

TEXTLINE vwchlist1 ( "VIEW CHECKLIST FOR CURRENT ASSESSMENT" )

BUTTON vwchlist {
    num_rows      1
    num_columns   42
    textline      vwchlist1, 0, 2
    helpfile      help/viewlist.hlp
}

-----
-   Declarations for majoraction footers
-----

```

```

WINDOW majoraction1 {      -header for problem status screen
    num_rows      4
    num_columns    78
    line          0, 0, 0.77
    title         "Alternative actions you can now take:"
    button        probdaf, 2, 2, "NEW_SCREEN probdafscren"
    button        analysis, 2,56, "NEW_SCREEN analysiscreen"
    button        vwchklist, 3, 2, "NEW_SCREEN viewchecklistscren"
    button        reportgen, 3,56, "CALL dummy2"
}

WINDOW majoraction1 {      -header for problem definition screen
    num_rows      4
    num_columns    78
    line          0, 0, 0.77
    title         "Alternative actions you can now take:"
    button        analysis, 2, 2, "NEW_SCREEN analysiscreen"
    button        probstat2, 2,48, "CALL peprobot"
    button        vwchklist, 3, 2, "NEW_SCREEN viewchecklistscren"
    button        reportgen, 3,48, "CALL dummy2"
}

WINDOW majoraction2 {      -header for data analysis screen
    num_rows      4
    num_columns    78
    line          0, 0, 0.77
    title         "Alternative actions you can now take:"
    button        probdaf1, 2, 2, "NEW_SCREEN probdafscren"
    button        reportgen, 2,47, "CALL dummy2"
    button        vwchklist, 3, 2, "NEW_SCREEN viewchecklistscren"
    button        probstat2, 3,47, "CALL peprobot"
}

WINDOW majoraction3 {      -header for report generation screen
    num_rows      4
    num_columns    78
    line          0, 0, 0.77
    title         "Alternative actions you can now take:"
    button        vwchklist, 2, 2, "NEW_SCREEN viewchecklistscren"
    button        probstat2, 2,47, "CALL peprobot"
    button        probdaf1, 3, 2, "NEW_SCREEN probdafscren"
    button        analysis, 3,47, "NEW_SCREEN analysiscreen"
}

WINDOW majoraction4 {      -header for view checklist screen
    num_rows      4
    num_columns    78
    line          0, 0, 0.77
    title         "Alternative actions you can now take:"
    button        probstat2, 2, 2, "CALL peprobot"
    button        reportgen, 2,47, "CALL dummy2"
    button        probdaf1, 3, 2, "NEW_SCREEN probdafscren"
    button        analysis, 3,47, "NEW_SCREEN analysiscreen"
}

TEXTLINE loadglobal ("LOAD LOCAL DATA FOR GENERAL ACCESS")

BUTTON loadglobal {
    num_rows      1
    num_columns    39
    textline      loadglobal, 0, 2
    helpfile      help/loadglob.hlp
}

WINDOW majoraction5 {      -footer for database housekeeping screen
    num_rows      4
    num_columns    78
    line          0, 0, 0.77
    title         "Alternative actions you can now take:"
    button        loadglobal, 2, 2, "CALL dummy"
    button        assessment, 3, 2, "CALL peprobot"
}

SCREEN mentry {
    title         "MTR DATA ENTRY"
    window        mtrmaincom, 3,1
    window        mtrdataentry, 7,1
    window        majoraction3, 10,1
    border        YES
}

```

```

)

-----
-      Declarations for problem status screen
-----

VARIABLE startdate {
    foundin "VARCHAR startdate.arr"
    type    STRING
    format  4-21s
}

DATUM startdate {
    num_rows    1
    num_columns 40
    variable    startdate, 0, 15
    leader      "Date started: "
    pickable    NO
}

VARIABLE lastdate {
    foundin "VARCHAR lastdate.arr"
    type    STRING
    format  4-21s
}

DATUM lastdate {
    num_rows    1
    num_columns 55
    variable    lastdate, 0, 28
    leader      "Date of last modification: "
    pickable    NO
}

VARIABLE planrlast {
    foundin "VARCHAR planrlast.arr"
    type    STRING
    format  4-20s
}

DATUM planrlast {
    num_rows    1
    num_columns 55
    variable    planrlast, 0, 19
    leader      "Last modified by: "
    pickable    NO
}

WINDOW probstat {
    num_rows    5
    num_columns 78
    datum       startdate, 0, 2
    datum       lastdate, 2, 2
    datum       planrlast, 4, 2
}

TEXTLINE chgasstxt ("Work on a different assessment")

BUTTON chgass {
    num_rows    1
    num_columns 70
    textline    chgasstxt, 0, 2
    helpfile    help/probdef.hlp
}

WINDOW chgasstxt {
    num_rows    1
    num_columns 78
    button      chgass, 0, 2, "CALL probstat"
}

```

```

-----
-      Declarations for ENVIRONMENTAL ASSESSMENT DEFINITION SCREEN
-----

TEXTLINE modmoetxt ("Work with MOA information (number or type of aircraft, missions, etc.)")

BUTTON modmoa {
    num_rows    1
    num_columns 76
    textline    modmoetxt, 0, 2
}

```

```

        helpfile    help/moswork.hlp
    }

TEXTLINE modstrtxt {"Work with MTR information (number or type of aircraft, missions, etc.)"}

BUTTON modstr {
    num_rows      1
    num_columns   76
    textline      modstrtxt, 0, 2
    helpfile      help/mtrwork.hlp
}

TEXTLINE modmaptxt {"Work with map information (designate land uses, update maps)"}

BUTTON modmap {
    num_rows      1
    num_columns   76
    textline      modmaptxt, 0, 2
    helpfile      help/mapwork.hlp
}

TEXTLINE selecttxt {"Actions you can now take to add information to this assessment:"}

WINDOW selection {
    num_rows      7
    num_columns   77
    textline      selecttxt, 0, 1
    button        modmap, 2, 1, "CALL dummy2"
    button        modstr, 4, 1, "CALL pantreat"
    button        modmap, 6, 1, "CALL dummy"
}

SCREEN probdefscreen {
    title "ENVIRONMENTAL ASSESSMENT DEFINITION"
    window assnecomm, 3, 1
    window selection, 7, 2
    window majoraction, 18, 1
    border YES
}

-----
- Declaration for entering coordinates window
-----

TEXTLINE entupplf
{"Enter upper-left corner coordinates of area of current interest"}

TEXTLINE entlowrt
{"Enter lower right corner coordinates of area of current interest"}

DATUM shvlat2 {
    num_rows      1
    num_columns   25
    variable      shvlat, 0, 11
    leader        "Latitude: "
    helpfile      help/combmap.hlp
}

DATUM shvlong2 {
    num_rows      1
    num_columns   25
    variable      shvlong, 0, 12
    leader        "Longitude: "
    helpfile      help/combmap.hlp
}

WINDOW entcoor {
    num_rows      10
    num_columns   78
    textline      entupplf, 2, 2
    datum         entlat, 3, 1, "CALL lat2dec test", "NEWVALS"
    datum         entlong, 3, 38, "CALL lon2dec test", "NEWVALS"
    textline      entlowrt, 5, 2
    datum         shvlat2, 6, 1, "CALL lat2dec ishow", "NEWVALS"
    datum         shvlong2, 6, 38, "CALL lon2dec ishow", "NEWVALS"
}

-----
- Declarations for DATA ANALYSIS SCREEN
-----

TEXTLINE geodatingtxt {"Make geodata inquiries on map screen"}

```

```

BUTTON geodating {
    num_rows 1
    num_columns 76
    textline geodatingtxt, 0,2
    helpfile help/geodating.hlp
}

TEXTLINE compnoisef { "Compare noise affects" } -1-17-88

BUTTON compnoisef {
    num_rows 1
    num_columns 76
    textline compnoiseftxt, 0,2
    helpfile help/compnoisef.hlp
} -1-17-88

TEXTLINE calcmoisef { "Calculate noise affects in specified area" }

BUTTON calcmoisef {
    num_rows 1
    num_columns 76
    textline calcmoiseftxt, 0,2
    helpfile help/affcalc.hlp
}

TEXTLINE calcmoisexp { "Calculate noise exposure in specified area" }

BUTTON calcmoisexp {
    num_rows 1
    num_columns 76
    textline calcmoisexp.txt, 0,2
    helpfile help/expcalc.hlp
}

TEXTLINE calqlklook { "Calculate quicklook (point) exposure estimate" }

BUTTON calqlklook {
    num_rows 1
    num_columns 50
    textline calqlklook.txt, 0,2
    helpfile help/qlklook.hlp
}

TEXTLINE entcoort { "Enter coordinates from keyboard" }

BUTTON entcoor {
    num_rows 1
    num_columns 40
    textline entcoortxt, 0,2
    helpfile help/kbdentry.hlp
}

TEXTLINE usemap { "Use map screen" }

BUTTON usemap {
    num_rows 1
    num_columns 30 textline usemap.txt, 0,2
    helpfile help/usemap.hlp
}

TEXTLINE selectanastxt { "Actions you can now take to analyze environmental assessment data:" }

TEXTLINE dafgeocarext { "Specify a geographic area of interest:" }

WINDOW dataaction {
    num_rows 11
    num_columns 76
    textline selectanastxt, 0,2
    button calqlklook, 2,2, "CALL dummy"
    button calcmoisexp, 3,2, "CALL dummy"
    button calcmoisef, 4,2, "CALL dummy"
    button compnoisef, 5,2, "CALL dummy"
    button geodating, 6,2, "CALL dummy"
    textline dafgeocarext, 8,2
    button usemap, 9,2, "CALL dummy2"
    button entcoor, 10,2, "ADD_WINDOW entcoor 5 1"
}

SCREEN analysiscreen {
    title "DATA ANALYSIS"
    window assnsmoon, 2,1
    window dataaction, 6,1

```



```

window majoraction2, 10 , 1
border YES
}

TEXTLINE showmoretxt ("Show more assessments (if any)")

BUTTON showmore {
    num_rows 1
    num_columns 37
    textline showmoretxt, 0, 2
    helpfile help/getassmnt.hlp
}

TEXTLINE recallasstxt ("Recall one of the following assessments:")

TEXTLINE stnewasstxt ("Start a new assessment")

BUTTON stnewass {
    num_rows 1
    num_columns 28
    textline stnewasstxt, 0, 2
    helpfile help/newassmnt.hlp
}

VARIABLE newassnm {
    foundin "VARCHAR n2bv.arr"
    type STRING
    format 4-30s
}

DATUM newassnm {
    num_rows 1
    num_columns 40
    variable newassnm, 0, 7
    loader "Name:"
    helpfile help/stnewass.hlp
}

WINDOW newassnm {
    num_rows 1
    num_columns 40
    datum newassnm, 0, 1
}

WINDOW stnewass {
    num_rows 1
    num_columns 78
    button stnewass, 0, 2, "ADD_WINDOW newassnm, 6 3",
        "UPDATE_DATUM newassnm",
        "CALL psnwassn n2bv.arr"
    button showmore, 0, 41, "CALL abunch"
}

WINDOW recallass {
    num_rows 1
    num_columns 78
    textline recallasstxt, 0, 2
}

WINDOW curasabut {
    num_rows 10
    num_columns 2
    button m1but00, 0, 0, "CALL ASANWoon deplmult[0].arr"
    button m1but01, 1, 0, "CALL ASANWoon deplmult[1].arr"
    button m1but02, 2, 0, "CALL ASANWoon deplmult[2].arr"
    button m1but03, 3, 0, "CALL ASANWoon deplmult[3].arr"
    button m1but04, 4, 0, "CALL ASANWoon deplmult[4].arr"
    button m1but05, 5, 0, "CALL ASANWoon deplmult[5].arr"
    button m1but06, 6, 0, "CALL ASANWoon deplmult[6].arr"
    button m1but07, 7, 0, "CALL ASANWoon deplmult[7].arr"
    button m1but08, 8, 0, "CALL ASANWoon deplmult[8].arr"
    button m1but09, 9, 0, "CALL ASANWoon deplmult[9].arr"
}

WINDOW curasabut1 {
    num_rows 10
    num_columns 2
    button m1but10, 0, 0, "CALL ASANWoon deplmult[10].arr"
    button m1but11, 1, 0, "CALL ASANWoon deplmult[11].arr"
    button m1but12, 2, 0, "CALL ASANWoon deplmult[12].arr"
    button m1but13, 3, 0, "CALL ASANWoon deplmult[13].arr"
    button m1but14, 4, 0, "CALL ASANWoon deplmult[14].arr"
}

```

```

button  subbut15, 5, 0, "CALL ASANocan deplmult[15].arr"
button  subbut16, 6, 0, "CALL ASANocan deplmult[16].arr"
button  subbut17, 7, 0, "CALL ASANocan deplmult[17].arr"
button  subbut18, 8, 0, "CALL ASANocan deplmult[18].arr"
button  subbut19, 9, 0, "CALL ASANocan deplmult[19].arr"

```

```

TEXTLINE retacass ("Continue without selecting assessment")

```

```

BUTTON retassent {
  num_rows 1
  num_columns 40
  textline retacass, 0, 2
  helpfile help/assassess.hlp
}

```

```

WINDOW retwocass {
  num_rows 1
  num_columns 76
  button retassent, 0, 1, "CALL ulists", "CALL pprobet"
}

```

```

SCREEN objcurasscreen {
  title "SELECT ANOTHER ASSESSMENT"
  window assanocan, 2, 1
  window staecass, 6, 1
  window recalass, 8, 1
  window curasabut, 9, 3
  window curasabut1, 9, 42
  window curdat, 9, 5
  window curdat1, 9, 44
  window retwocass, 20, 2
  border YES
}

```

Declarations for NEW ASSESSMENT DEFINITION

```

VARIABLE entdesc1 {
  foundin "VARCHAR60 entdesc[0].arr"
  type STRING
  format %60s
}

```

```

DATUM entdesc1 {
  num_rows 1
  num_columns 75
  variable entdesc1, 0, 0
  helpfile help/entdesc.hlp
}

```

```

VARIABLE entdesc2 {
  foundin "VARCHAR60 entdesc[1].arr"
  type STRING
  format %60s
}

```

```

DATUM entdesc2 {
  num_rows 1
  num_columns 75
  variable entdesc2, 0, 0
  helpfile help/entdesc.hlp
}

```

```

VARIABLE entdesc3 {
  foundin "VARCHAR60 entdesc[2].arr"
  type STRING
  format %60s
}

```

```

DATUM entdesc3 {
  num_rows 1
  num_columns 75
  variable entdesc3, 0, 0
  helpfile help/entdesc.hlp
}

```

```

VARIABLE entdesc4 {
  foundin "VARCHAR60 entdesc[3].arr"
  type STRING
  format %60s
}

```

```

    )

    DATUM   entdesc4 {
        num_rows 1
        num_columns 75
        variable   entdesc4, 0,0
        helpfile   help/entdesc.hlp
    }

    TEXTLINE entdesc4txt ("Please enter a brief description for this assessment")

    DATUM   newassnam {
        num_rows 1
        num_columns 77
        variable   newassnam, 0,35
        leader     "Name of new assessment definition:"
        pinkable   NO
    }

    WINDOW newassdesc {
        num_rows 9
        num_columns 70
        datum      newassnam, 1,3
        textline    entdesc4txt, 3,4
        datum      entdesc4, 4,4
        datum      entdesc2, 5,4
        datum      entdesc3, 6,4
        datum      entdesc4, 7,4
        border      YES
    }

    -----
    ~           Declarations for database housekeeping screen
    -----

    TEXTLINE updatainfotxt ("Update information")

    BUTTON   updatainfo {
        num_rows 1
        num_columns 50
        textline updatainfotxt, 0,2
        helpfile help/updatain.hlp
    }

    TEXTLINE asstabletxt2 ("Print list of all columns in an assessment's tables")

    BUTTON   asstable2 {
        num_rows 1
        num_columns 55
        textline asstabletxt2, 0,2
        helpfile help/asstable.hlp
    }

    TEXTLINE asstabletxt1 ("Print list of an assessment's tables")

    BUTTON   asstable1 {
        num_rows 1
        num_columns 50
        textline asstabletxt1, 0,2
        helpfile help/asstable.hlp
    }

    TEXTLINE assessetxt ("Print list of all assessments")

    BUTTON   assess {
        num_rows 1
        num_columns 50
        textline assessetxt, 0,2
        helpfile help/asshelp.hlp
    }

    TEXTLINE dbdatetxt ("Print all database dates")

    BUTTON   dbdata {
        num_rows 1
        num_columns 50
        textline dbdatetxt, 0,2
        helpfile help/dbdata.hlp
    }

    TEXTLINE dbshakpgtxt ("WARNING: Actions you take on this screen affect ASAM'S
        permanent databases!")

```

```

WINDOW dbhsakpgaction {
    num_rows      13
    num_columns    78
    line          0,0, 0,77
    textline      dbhsakpgtxt, 2,2
    button        dbdate, 4,2, "CALL dummy"
    button        asdesc, 6,2, "CALL prntabc"
    button        asstable1, 8,2, "CALL $Upropt 1"
    button        asstable2, 10,2, "CALL $Upropt 2"
    button        updtainfo, 12,2, "CALL dummy"
}

SCREEN dbhsakpgscreen {
    title         "DATABASE HOUSEKEEPING"
    window        dbhsakpgaction, 2,1
    window        majoraction3, 18,1
    border        YES
}

WINDOW quickasabut {
    num_rows      10
    num_columns    2
    button        subut00, 0, 0, "CALL $Uprint deplmult[0].arr"
    button        subut01, 1, 0, "CALL $Uprint deplmult[1].arr"
    button        subut02, 2, 0, "CALL $Uprint deplmult[2].arr"
    button        subut03, 3, 0, "CALL $Uprint deplmult[3].arr"
    button        subut04, 4, 0, "CALL $Uprint deplmult[4].arr"
    button        subut05, 5, 0, "CALL $Uprint deplmult[5].arr"
    button        subut06, 6, 0, "CALL $Uprint deplmult[6].arr"
    button        subut07, 7, 0, "CALL $Uprint deplmult[7].arr"
    button        subut08, 8, 0, "CALL $Uprint deplmult[8].arr"
    button        subut09, 9, 0, "CALL $Uprint deplmult[9].arr"
}

WINDOW quickasabut1 {
    num_rows      10
    num_columns    2
    button        subut10, 0, 0, "CALL $Uprint deplmult[10].arr"
    button        subut11, 1, 0, "CALL $Uprint deplmult[11].arr"
    button        subut12, 2, 0, "CALL $Uprint deplmult[12].arr"
    button        subut13, 3, 0, "CALL $Uprint deplmult[13].arr"
    button        subut14, 4, 0, "CALL $Uprint deplmult[14].arr"
    button        subut15, 5, 0, "CALL $Uprint deplmult[15].arr"
    button        subut16, 6, 0, "CALL $Uprint deplmult[16].arr"
    button        subut17, 7, 0, "CALL $Uprint deplmult[17].arr"
    button        subut18, 8, 0, "CALL $Uprint deplmult[18].arr"
    button        subut19, 9, 0, "CALL $Uprint deplmult[19].arr"
}

TEXTLINE suselact {"Print list of SUPERUSER's tables"}
TEXTLINE hqselact {"Print list of HEADQUARTERS' tables"}

BUTTON suselact {
    num_rows      1
    num_columns    50
    textline      suselact, 0,2
    helpfile      help/nohelp.hlp
}

BUTTON hqselact {
    num_rows      1
    num_columns    50
    textline      hqselact, 0,2
    helpfile      help/nohelp.hlp
}

WINDOW retwopr {
    num_rows      1
    num_columns    78
    button        retasent, 0,1, "CALL ulists", "NEW_SCREEN dbhsakpgscreen"
}

WINDOW hqsuselact {
    num_rows      3
    num_columns    78
    button        hqselact, 0,1, "CALL $Upropt 3"
    button        suselact, 2,1, "CALL $Upropt 4"
}

SCREEN sloascreen {

```

```

title      "SELECT ASSESSMENT FOR DATABASE PRINTOUT"
window    hqsselect,  4,1
window    recclass,   8,1
window    quickasabut, 9,3
window    quickasabut1,9,42
window    curdat,     9,5
window    curdat1,    9,44
window    netwoprn,   20,2
border    YES
}

```

 " Declarations for view CHECK-LIST SCREEN

```
TEXTLINE otherchklist ("View another checklist")
```

```

BUTTON otherchklist (
  num_rows      1
  num_columns   25
  textline      otherchklist, 0, 2
  helpfile      help/otherchk.hlp
)

```

```

TEXTBLOCK fcas12 (
  filename      trtblk/fcas12.txt
  num_rows      10
  num_columns   76
  border        YES
)

```

```

WINDOW fcas12 (
  num_rows      13
  num_columns   78
  textblock     fcas12, 0,1
  button        otherchklist, 12,2, "REMOVE_WINDOW"
)

```

```

TEXTBLOCK fcas11 (
  filename      trtblk/fcas11.txt
  num_rows      10
  num_columns   76
  border        YES
)

```

```

WINDOW fcas11 (
  num_rows      13
  num_columns   78
  textblock     fcas11, 0,1
  button        otherchklist, 12,2, "REMOVE_WINDOW"
)

```

```

TEXTBLOCK ostar3 (
  filename      trtblk/ostar3.txt
  num_rows      10
  num_columns   76
  border        YES
)

```

```

WINDOW ostar3 (
  num_rows      13
  num_columns   78
  textblock     ostar3, 0,1
  button        otherchklist, 12,2, "REMOVE_WINDOW"
)

```

```

TEXTBLOCK ostar2 (
  filename      trtblk/ostar2.txt
  num_rows      10
  num_columns   76
  border        YES
)

```

```

WINDOW ostar2 (
  num_rows      13

```

```

num_columns      78
textblock        catx2, 0,1
button           otherchklist, 12,2, "REMOVE_WINDOW"
}

TEXTBLOCK catx1 {
    filename      txtblk/catx1.txt
    num_rows      10
    num_columns    76
    border        YES
}

WINDOW catx1 {
    num_rows      13
    num_columns    78
    textblock      catx1, 0,1
    button         otherchklist, 12,2, "REMOVE_WINDOW"
}

TEXTLINE fonsi2txt {
    "Show documentation necessary for finding of no significant impact"}

BUTTON fonsi2 {
    num_rows      1
    num_columns    72
    textline       fonsi2txt, 0, 2
    helpfile       help/fonsi2.hlp
}

TEXTLINE fonsi1txt {
    "Show NEPA bases for finding of no significant impact (FONSI)"}

BUTTON fonsi1 {
    num_rows      1
    num_columns    65
    textline       fonsi1txt, 0, 2
    helpfile       help/fonsi1.hlp
}

TEXTLINE catx3txt {"Show documentation needed for categorical exclusions"}

BUTTON catx3 {
    num_rows      1
    num_columns    55
    textline       catx3txt, 0, 2
    helpfile       help/catx3.hlp
}

TEXTLINE catx2txt {
    "Show examples of proposed actions qualifying for categorical exclusions"}

BUTTON catx2 {
    num_rows      1
    num_columns    74
    textline       catx2txt, 0, 2
    helpfile       help/catx2.hlp
}

TEXTLINE catx1txt {"Show NEPA bases for categorical exclusions (CATX)"}

BUTTON catx1 {
    num_rows      1
    num_columns    55
    textline       catx1txt, 0, 2
    helpfile       help/catx1.hlp
}

WINDOW vwbklist {
    num_rows      11
    num_columns    78
    line          0,0 , 0,77
    button         catx1,      2,2, "ADD_WINDOW catx1 3 1"
    button         catx2,      4,2, "ADD_WINDOW catx2 3 1"
    button         catx3,      6,2, "ADD_WINDOW catx3 3 1"
    button         fonsi1,     8,2, "ADD_WINDOW fonsi1 3 1"
    button         fonsi2,    10,2, "ADD_WINDOW fonsi2 3 1"
}

SCREEN viewchecklistscreen {
    title          "VIEW CHECKLIST"
    window         vwbklist, 2,1
    window         majoraction4, 18,1

```

```

border    YES
}

-----

SCREEN probatatscreen (
  title    "ENVIRONMENTAL ASSESSMENT STATUS"
  mainscreen YES
  border    YES
  window    assemmoom,      3,1
  window    probatstat,     7,1
  window    chgassess,      16,1
  window    majoaction,     18,1
)

-END OF ASAM.SDT

```

```

-INCLUDE STATEMENT FOR THE ASAM typedef DEFINITIONS NEEDED TO
-  COMPILER USING C WITHOUT INCLUDING THE NAME OF THE COMPILER

```

```

-----
INCLUDE ASAMTYPE.H
INCLUDE ASAM.H

```

```

-----
- DECLARATIONS IN NON-LEXICAL OBJECT SEQUENCE .....
-----

```

```

- Multiple Choice Space
-----

```

```

VARIABLE maldat00 (
  foundin "VARCHAR30 deplmult[0].arr"
  type    STRING
  format  4-30s
)

DATUM maldat00 (
  num_rows    1
  num_columns  32
  variable    maldat00, 0, 2
  pickable    NO
)

VARIABLE maldat01 (
  foundin "VARCHAR30 deplmult[1].arr"
  type    STRING
  format  4-30s
)

DATUM maldat01 (
  num_rows    1
  num_columns  32
  variable    maldat01, 0, 2
  pickable    NO
)

VARIABLE maldat02 (
  foundin "VARCHAR30 deplmult[2].arr"
  type    STRING
  format  4-30s
)

DATUM maldat02 (
  num_rows    2
  num_columns  32
  variable    maldat02, 0, 2
  pickable    NO
)

VARIABLE maldat03 (
  foundin "VARCHAR30 deplmult[3].arr"
  type    STRING
  format  4-30s
)

DATUM maldat03 (

```

```

num_rows 1
num_columns 32
variable maldat03, 0, 2
pickable NO
}

VARIABLE maldat04 {
  foundin "VARCHAR30 deplmult[4].arr"
  type STRING
  format t-30s
}

DATUM maldat04 {
  num_rows 1
  num_columns 32
  variable maldat04, 0, 2
  pickable NO
}

VARIABLE maldat05 {
  foundin "VARCHAR30 deplmult[5].arr"
  type STRING
  format t-30s
}

DATUM maldat05 {
  num_rows 1
  num_columns 32
  variable maldat05, 0, 2
  pickable NO
}

VARIABLE maldat06 {
  foundin "VARCHAR30 deplmult[6].arr"
  type STRING
  format t-30s
}

DATUM maldat06 {
  num_rows 1
  num_columns 32
  variable maldat06, 0, 2
  pickable NO
}

VARIABLE maldat07 {
  foundin "VARCHAR30 deplmult[7].arr"
  type STRING
  format t-30s
}

DATUM maldat07 {
  num_rows 1
  num_columns 32
  variable maldat07, 0, 2
  pickable NO
}

VARIABLE maldat08 {
  foundin "VARCHAR30 deplmult[8].arr"
  type STRING
  format t-30s
}

DATUM maldat08 {
  num_rows 1
  num_columns 32
  variable maldat08, 0, 2
  pickable NO
}

VARIABLE maldat09 {
  foundin "VARCHAR30 deplmult[9].arr"
  type STRING
  format t-30s
}

DATUM maldat09 {
  num_rows 1
  num_columns 32
  variable maldat09, 0, 2
  pickable NO
}

```



```

    )

    VARIABLE maldat10 {
        foundin "VARCHAR30 deplmult[10].arr"
        type    STRING
        format   4-30s
    }

    DATUM maldat10 {
        num_rows    1
        num_columns  32
        variable     maldat10, 0, 2
        pickable     NO
    }

    VARIABLE maldat11 {
        foundin "VARCHAR30 deplmult[11].arr"
        type    STRING
        format   4-30s
    }

    DATUM maldat11 {
        num_rows    1
        num_columns  32
        variable     maldat11, 0, 2
        pickable     NO
    }

    VARIABLE maldat12 {
        foundin "VARCHAR30 deplmult[12].arr"
        type    STRING
        format   4-30s
    }

    DATUM maldat12 {
        num_rows    1
        num_columns  32
        variable     maldat12, 0, 2
        pickable     NO
    }

    VARIABLE maldat13 {
        foundin "VARCHAR30 deplmult[13].arr"
        type    STRING
        format   4-30s
    }

    DATUM maldat13 {
        num_rows    1
        num_columns  32
        variable     maldat13, 0, 2
        pickable     NO
    }

    VARIABLE maldat14 {
        foundin "VARCHAR30 deplmult[14].arr"
        type    STRING
        format   4-30s
    }

    DATUM maldat14 {
        num_rows    1
        num_columns  32
        variable     maldat14, 0, 2
        pickable     NO
    }

    VARIABLE maldat15 {
        foundin "VARCHAR30 deplmult[15].arr"
        type    STRING
        format   4-30s
    }

    DATUM maldat15 {
        num_rows    1
        num_columns  32
        variable     maldat15, 0, 2
        pickable     NO
    }

    VARIABLE maldat16 {
        foundin "VARCHAR30 deplmult[16].arr"

```

```

    type    STRING
    format  4-30s
}

DATUM maldet16 {
    num_rows    1
    num_columns 32
    variable    maldet16, 0, 2
    pickable    NO
}

VARIABLE maldet17 {
    foundin "VARCHAR30 deplmult[17].arr"
    type    STRING
    format  4-30s
}

DATUM maldet17 {
    num_rows    1
    num_columns 32
    variable    maldet17, 0, 2
    pickable    NO
}

VARIABLE maldet18 {
    foundin "VARCHAR30 deplmult[18].arr"
    type    STRING
    format  4-30s
}

DATUM maldet18 {
    num_rows    1
    num_columns 32
    variable    maldet18, 0, 2
    pickable    NO
}

VARIABLE maldet19 {
    foundin "VARCHAR30 deplmult[19].arr"
    type    STRING
    format  4-30s
}

DATUM maldet19 {
    num_rows    1
    num_columns 32
    variable    maldet19, 0, 2
    pickable    NO
}

- Planner attributes
-----

VARIABLE planname {
    foundin "VARCHAR planname.arr"
    type    STRING
    format  4-30s
}

DATUM planname {
    num_rows    1
    num_columns 50
    variable    planname, 0, 20
    leader      "Your name, please:"
    helpfile    help/username.hlp
}

VARIABLE password {
    type    STRING
    format  4-10s
}

DATUM password {
    num_rows    1
    num_columns 55
    variable    password, 0, 23
    leader      "Please enter password:"
    helpfile    help/password.hlp
}

- Other General Storage for Verification, Tests, Etc.
-----

```

```

VARIABLE mstrmtr {
    foundin "VARCHAR mstrv.arr"
    type     STRING
    format   4-20s
}

- Noise Source Attributes
- -----

VARIABLE srcid {
    foundin "VARCHAR srcid.arr"
    type     STRING
    format   4-9s
}

DATUM  mtrmtr {
    num_rows      1
    num_columns    60
    variable       srcid, 0, 21
    leader         "Name of current MTR:"
    pickable       NO
}

VARIABLE srodesc {
    foundin "VARCHAR srodesc.arr"
    type     STRING
    format   4-54s
}

DATUM  srods {
    num_rows      1
    num_columns    76
    variable       srodesc, 0, 22
    leader         "Description:"
    helpfile       help/mtrdesc.hlp
}

DATUM  mtrdesc {
    num_rows      1
    num_columns    70
    variable       srodesc, 0, 7
    leader         "Note: "
    pickable       NO
}

VARIABLE datapub1 {
    foundin "VARCHAR srodate.arr"
    type     STRING
    format   4-12s
}

VARIABLE schedule {
    foundin "VARCHAR srosched.arr"
    type     STRING
    format   4-50s
}

DATUM  schedule {
    num_rows      1
    num_columns    73
    variable       schedule, 0, 22
    leader         "Scheduling activity:"
    helpfile       help/schdtr.hlp
}

VARIABLE sroorig {
    foundin "VARCHAR sroorig.arr"
    type     STRING
    format   4-80s
}

DATUM  origmtr {
    num_rows      1
    num_columns    73
    variable       sroorig, 0, 22
    leader         "Originating activity:"
    helpfile       help/origmtr.hlp
}

- MTR Attributes
- -----

```

```

VARIABLE curartoc {
    foundin "VARCHAR curartoc.arr"
    type STRING
    format %3s
}

DATUM curartoc {
    num_rows 1
    num_columns 27
    variable curartoc, 0,16
    leader "ARTOC: "
    helpfile help/curartoc.hlp
}

VARIABLE curwidright {
    type INTEGER
    format %2d
}

DATUM curwidright {
    num_rows 1
    num_columns 25
    variable curwidright, 0,16
    leader "Width (right): "
    helpfile help/curwidrt.hlp
}

VARIABLE curwidleft {
    type INTEGER
    format %2d
}

DATUM curwidleft {
    num_rows 1
    num_columns 25
    variable curwidleft, 0,16
    leader "Width (left): "
    helpfile help/curwidlf.hlp
}

VARIABLE curhighalt {
    foundin "ALTSPEC curhighalt.spec"
    type STRING
    format %3s
}

DATUM curhighalt {
    num_rows 1
    num_columns 27
    variable curhighalt, 0,16
    leader "High altitude: "
    helpfile help/curhialt.hlp
}

VARIABLE curlowalt {
    foundin "ALTSPEC curlowalt.spec"
    type STRING
    format %3s
}

VARIABLE preartoc {
    foundin "VARCHAR preartoc.arr"
    type STRING
    format %3s
}

DATUM preartoc {
    num_rows 1
    num_columns 8
    variable preartoc, 0,4
    pickable NO
}

VARIABLE prewidright {
    type INTEGER
    format %2d
}

DATUM prewidright {
    num_rows 1
    num_columns 8
    variable prewidright, 0,4

```

- CURRENT Set

- PREVIOUS Set

```

        pickable      NO
    }

VARIABLE prwidleft {
    type      INTEGER
    format    %2d
}

DATUM prwidleft {
    num_rows      1
    num_columns    0
    variable      prwidleft, 0,4
    pickable      NO
}

VARIABLE prhighalt {
    foundin "ALTSPEC prhighalt.spec"
    type    STRING
    format  %9s
}

DATUM prhighalt {
    num_rows      1
    num_columns    10
    variable      prhighalt, 0,1
    pickable      NO
}

VARIABLE prlowalt {
    foundin "ALTSPEC prlowalt.spec"
    type    STRING
    format  %9s
}

- Coordinates
- -----

VARIABLE entlat {
    foundin "COORDINATE ent.lat"
    type    STRING
    format  %13s
}
- CURRENT or ENTER Set

VARIABLE entlong {
    foundin "COORDINATE ent.long"
    type    STRING
    format  %13s
}

VARIABLE shwlat {
    foundin "COORDINATE shw.lat"
    type    STRING
    format  %13s
}
- PREVIOUS or SHOW Set

VARIABLE shwlong {
    foundin "COORDINATE shw.long"
    type    STRING
    format  %13s
}

- Navigation Point Properties
- -----

VARIABLE curfixtyp {
    foundin "VARIABLE curfixtyp.arr"
    type    STRING
    format  %12s
}
- CURRENT or ENTER Set

VARIABLE curfindist {
    type    INTEGER
    format  %3d
}

VARIABLE curfixrad {
    type    INTEGER
    format  %3d
}

VARIABLE curfixid {
    foundin "VARIABLE curfixid.arr"

```

```

        type      STRING
        format    %5s
    }

    VARIABLE curnavpt {
        foundin   "VARCHAR curnavpt.arr"
        type      STRING
        format    %3s
    }

    VARIABLE prefixtyp {
        foundin   "VARCHAR prefixtyp.arr"
        type      STRING
        format    %12s
    }
    - PREVIOUS or DISPLAY Set

    VARIABLE prefixdist {
        type      INTEGER
        format    %3d
    }

    VARIABLE prefixrad {
        type      INTEGER
        format    %3d
    }

    VARIABLE prefixid {
        foundin   "VARCHAR prefixid.arr"
        type      STRING
        format    %5s
    }

    VARIABLE prenavpt {
        foundin   "VARCHAR prenavpt.arr"
        type      STRING
        format    %3s
    }

    - Aircraft Parameters
    -----

    VARIABLE acname {
        foundin   "VARCHAR ac_name.arr"
        type      STRING
        format    %-12s
    }

    DATUM acname {
        num_rows 1
        num_columns 25
        variable  acname, 0, 10
        leader    "Aircraft:"
        pickable  NO
    }

    VARIABLE acname1 {
        foundin   "VARCHAR tid.arr"
        type      STRING
        format    %-12s
    }

    DATUM acname1 {
        num_rows 1
        num_columns 35
        variable  acname1, 0, 16
        leader    "Aircraft name:"
        helpfile  help/utrainrur.hlp
    }

    - Operations Specification
    -----

    VARIABLE day {
        foundin   "OPERATIONS ops[0].day"
        type      INTEGER
        format    %4d
    }
    - GENERIC or JANUARY

    VARIABLE janday {
        foundin   "OPERATIONS ops[0].day"
        type      INTEGER
        format    %4d
    }

```

```

    }

    VARIABLE night {
        foundin "OPERATIONS ops[0].nite"
        type INTEGER
        format %4d
    }

    VARIABLE jannite {
        foundin "OPERATIONS ops[0].nite"
        type INTEGER
        format %4d
    }

    VARIABLE febday {
        foundin "OPERATIONS ops[1].day"
        type INTEGER
        format %4d
    }

    VARIABLE febite {
        foundin "OPERATIONS ops[1].nite"
        type INTEGER
        format %4d
    }

    VARIABLE marday {
        foundin "OPERATIONS ops[2].day"
        type INTEGER
        format %4d
    }

    VARIABLE marnite {
        foundin "OPERATIONS ops[2].nite"
        type INTEGER
        format %4d
    }

    VARIABLE sprday {
        foundin "OPERATIONS ops[3].day"
        type INTEGER
        format %4d
    }

    VARIABLE sprnite {
        foundin "OPERATIONS ops[3].nite"
        type INTEGER
        format %4d
    }

    VARIABLE mayday {
        foundin "OPERATIONS ops[4].day"
        type INTEGER
        format %4d
    }

    VARIABLE maynite {
        foundin "OPERATIONS ops[4].nite"
        type INTEGER
        format %4d
    }

    VARIABLE junday {
        foundin "OPERATIONS ops[5].day"
        type INTEGER
        format %4d
    }

    VARIABLE junnite {
        foundin "OPERATIONS ops[5].nite"
        type INTEGER
        format %4d
    }

    VARIABLE julday {
        foundin "OPERATIONS ops[6].day"
        type INTEGER
        format %4d
    }

    VARIABLE jurnite {
        foundin "OPERATIONS ops[6].nite"

```

```

        type      INTEGER
        format    %4d
    )

    VARIABLE sepday {
        foundin   "OPERATIONS ops[7].day"
        type      INTEGER
        format    %4d
    }

    VARIABLE sepnite {
        foundin   "OPERATIONS ops[7].nite"
        type      INTEGER
        format    %4d
    }

    VARIABLE sepday {
        foundin   "OPERATIONS ops[8].day"
        type      INTEGER
        format    %4d
    }

    VARIABLE sepnite {
        foundin   "OPERATIONS ops[8].nite"
        type      INTEGER
        format    %4d
    }

    VARIABLE octday {
        foundin   "OPERATIONS ops[9].day"
        type      INTEGER
        format    %4d
    }

    VARIABLE octnite {
        foundin   "OPERATIONS ops[9].nite"
        type      INTEGER
        format    %4d
    }

    VARIABLE novday {
        foundin   "OPERATIONS ops[10].day"
        type      INTEGER
        format    %4d
    }

    VARIABLE novnite {
        foundin   "OPERATIONS ops[10].nite"
        type      INTEGER
        format    %4d
    }

    VARIABLE decday {
        foundin   "OPERATIONS ops[11].day"
        type      INTEGER
        format    %4d
    }

    VARIABLE decnite {
        foundin   "OPERATIONS ops[11].nite"
        type      INTEGER
        format    %4d
    }

- Missions
- -----

    VARIABLE misname {
        foundin   "VARCHAR misslabl.arr"
        type      STRING
        format    %-7s
    }

    DATUM misname1 {
        num_rows   1
        num_columns 34
        variable    misname, 0, 26
        loader      "Name of current mission: "
        pickable    NO
    }

    VARIABLE newmisname {

```



```

        foundia      "VARCHAR aid.aid"
        type         STRING
        format       4-7s
    }

    DATUM misname {
        num_rows      1
        num_columns    14
        variable       newmisname, 0, 14
        leader         "Mission name:"
        helpfile       help/misname.hlp
    }

    TEXTLINE select {
        "Right now you can type ? for help, <CTRL> C to quit, or move the cursor"
    TEXTLINE canvass {"CONDUCT AN ENVIRONMENTAL ASSESSMENT" }
    TEXTLINE housekp {"PERFORM DATABASE HOUSEKEEPING" }
    TEXTLINE introtxt {"VIEW GENERAL INFORMATION ABOUT THIS PROGRAM"}

    TEXTLINE selacmstrxt {"Select aircraft and mission for MFR"}
    TEXTLINE modcmstrxt {"Modify current MFR"}
    TEXTLINE chgmstrxt {"Select another MFR"}

    TEXTLINE nextnavptxt {"Enter next navigation point"}
    TEXTLINE curpretxt {
        { "CURRENT      PREVIOUS          CURRENT      PREVIOUS" }
        TEXTLINE curpretxt3
        { "-----      -----      -----      -----" }
    TEXTLINE cancalstrxt {"Cancel this MFR data entry"}
    TEXTLINE savemstrxt {"Save this MFR"}

    TEXTLINE showmorestrxt {"Show more MFR names (if any)"}
    TEXTLINE recallstrxt {"Recall one of the following MFRs:"}
    TEXTLINE stnewstrxt {"Start new MFR "}
    TEXTLINE definamstrxt {"Enter route waypoints"}

    TEXTLINE actxt {"Aircraft:"}

    TEXTLINE opseasontxt {"Operations are seasonal"}
    TEXTLINE opthrytxt {"Operations are even throughout year"}
    TEXTLINE instrtxt {"Please enter day and night operations by month"}
    TEXTLINE daynite {" DAY NIGHT DAY NIGHT DAY NIGHT DAY NIGHT"}
    TEXTLINE instrtxt {"Please enter daytime and night operations per month"}
    TEXTLINE cancalstrxt {"Abandon this mission"}
    TEXTLINE entaisdstrxt {"Enter this mission into database"}
    TEXTLINE stnewmstrxt {"Start new mission"}

    TEXTBLOCK introtxt {
        num_rows      14
        num_columns    76
        border         YES
        filename       txtblk/intro.txt
    }

    BUTTON assessment {
        num_rows      1
        num_columns    36
        textline       canvass, 0, 2
        helpfile       help/assess.hlp
    }

    BUTTON chgmstr {
        num_rows      1
        num_columns    35
        textline       chgmstrxt, 0, 2
        helpfile       help/calstr.hlp
    }

    BUTTON housekeeping {
        num_rows      1
        num_columns    35
        textline       housekp, 0, 2
        helpfile       help/housekp.hlp
    }

    BUTTON introtxt {
        num_rows      1
        num_columns    56
        textline       introtxt, 0, 2
        helpfile       help/introhel.hlp
    }

```

```

BUTTON modcumtr {
    num_rows 1
    num_columns 35
    textline modcumtrtxt, 0, 2
    helpfile help/modhelp.hlp
    - no help here cuz this button doesn't do anything
}

WINDOW password {
    num_rows 1
    num_columns 60
    datum password, 0, 5
}

TEXTLINE tit1txt
{"Developed for Noise and Sonic Boom Impact Technology Program"}

TEXTLINE tit2txt {"under U.S. Air Force Contract F33615-86-C-0530"}

TEXTLINE tit3txt {"by RSN Laboratories, Inc."}

TEXTLINE tit4txt {"February, 1988"}

TEXTLINE tit5txt
{"Unreleased demonstration of Prototype Version...Not for General Use"}

TEXTLINE diafo {"Does viewing general information on ASAN"}

BUTTON diafo {
    num_rows 1
    num_columns 76
    textline diafo, 0, 2
}

WINDOW introduction {
    num_rows 15
    num_columns 78
    textblock introtxt, 0, 1
    button diafo, 14, 1, "REMOVE_WINDOW"
}

WINDOW introwindow1 {
    num_rows 21
    num_columns 78
    line 1, 0, 1, 77
    title "ASSESSMENT SYSTEM FOR AIRCRAFT NOISE (ASAN)"
    textline tit1txt, 2, 9
    textline tit2txt, 3, 16
    textline tit3txt, 4, 27
    textline tit4txt, 5, 32
    textline tit5txt, 7, 6
    line 8, 0, 9, 77
    datum p1aarnam, 12, 1,
        "CALL VCAPITAL p1aarnam",
        "ADD_WINDOW password 15 3",
        "UPDATE DATUM password",
        "REMOVE_WINDOW",
        "CALL pwchack"
}

WINDOW introwindow2 {
    num_rows 9
    num_columns 78
    line 5, 0, 5, 77
    textline selact, 6, 3
    button introtxt, 7, 15, "ADD_WINDOW introduction 3 1"
    button assessment, 8, 2, "CALL pprobet"
    button housekeeping, 8, 44, "CALL pedbheak"
}

SCREEN firstscreen {
    titlescreen YES
    border YES
    title "ASSESSMENT SYSTEM FOR AIRCRAFT NOISE (ASAN)"
    window introwindow1, 1, 1
    window introwindow2, 13, 1
}

BUTTON opseascon {
    num_rows 3
    num_columns 41
    textline opseascontxt, 1, 3
}

```

```

border      YES
helpfile    help/vxblopc.hlp
}

BUTTON opsteady {
  num_rows  3
  num_columns 41
  textline   opthruyrtxt, 1, 3
  border     YES
  helpfile   help/evanops.hlp
}

BUTTON malbet00 {
  num_rows  1
  num_columns 1
  helpfile   help/malbet.hlp
}

BUTTON malbet01 {
  num_rows  1
  num_columns 1
  helpfile   help/malbet.hlp
}

BUTTON malbet02 {
  num_rows  1
  num_columns 1
  helpfile   help/malbet.hlp
}

BUTTON malbet03 {
  num_rows  1
  num_columns 1
  helpfile   help/malbet.hlp
}

BUTTON malbet04 {
  num_rows  1
  num_columns 1
  helpfile   help/malbet.hlp
}

BUTTON malbet05 {
  num_rows  1
  num_columns 1
  helpfile   help/malbet.hlp
}

BUTTON malbet06 {
  num_rows  1
  num_columns 1
  helpfile   help/malbet.hlp
}

BUTTON malbet07 {
  num_rows  1
  num_columns 1
  helpfile   help/malbet.hlp
}

BUTTON malbet08 {
  num_rows  1
  num_columns 1
  helpfile   help/malbet.hlp
}

BUTTON malbet09 {
  num_rows  1
  num_columns 1
  helpfile   help/malbet.hlp
}

BUTTON malbet10 {
  num_rows  1
  num_columns 2
  helpfile   help/malbet.hlp
}

BUTTON malbet11 {
  num_rows  1
  num_columns 2
  helpfile   help/malbet.hlp
}

```

```

    }

    BUTTON mabut12 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut13 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut14 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut15 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut16 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut17 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut18 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

    BUTTON mabut19 {
        num_rows 1
        num_columns 2
        helpfile help/mabut.hlp
    }

-----
-      Declarations for MTR DATA ENTRY SCREEN
-----

    BUTTON staemis {
        num_rows 1
        num_columns 40
        textline staemistxt, 0, 2
        helpfile help/staemis.hlp
    }

    BUTTON salacmis {
        num_rows 1
        num_columns 40
        textline salacmistxt, 0, 2
        helpfile help/salacmis.hlp
    }

    TEXTLINE specnumistxt ("Specify new mission")

    BUTTON specnumis {
        num_rows 1
        num_columns 30
        textline specnumistxt, 0, 2
        helpfile help/newmiss.hlp
    }

    WINDOW mtrdataentry {
        num_rows 8
        num_columns 78
        button chgmtr, 0, 2, "CALL pchgptr"
        button salacmis, 2, 2, "CALL pmtrmis"
    }

```

```

button      modcounttr, 4, 2, "CALL dummy"
button      specwarnis, 6, 2, "ADD WINDOW newmisam 13 3",
           "UPDATE DATUM misame",
           "CALL WHEREAMI Screen Window Datum Button",
           "CALL stropy oldscreen Screen",
           "CALL penwarnis aid.arr"

```

```

}

```

```

WINDOW mtrasmcom {
  num_rows      3
  num_columns    78
  line          2,0, 2,77
  datum         mtrasm, 0, 2
  datum         mtrdasc, 1, 2
}

```

```

WINDOW mtrasmcom1 {
  num_rows      4
  num_columns    78
  line          3,0, 3,77
  datum         mtrasm, 0, 2
  datum         mtrdasc, 1, 2
  datum         misame1, 2, 2
}

```

```

-----
-      Declarations for DEFINE/MODIFY SCREEN      (defmodstrscreen)
-----

```

```

DATUM prelowalt {
  num_rows      1
  num_columns    10
  variable       prelowalt, 0,1
  pickable      NO
}

```

```

DATUM curlowalt {
  num_rows      1
  num_columns    29
  variable       curlowalt, 0, 16
  leader         "Low altitude: "
  helpfile       help/curloalt.hlp
}

```

```

DATUM prefistype {
  num_rows      1
  num_columns    13
  variable       prefistyp, 0,0
  pickable      NO
}

```

```

DATUM curfistype {
  num_rows      1
  num_columns    24
  variable       curfistyp, 0,11
  leader         "Fix type: "
  helpfile       help/curfistp.hlp
}

```

```

-----
-      Declarations for entering coordinates
-----

```

```

DATUM entlong {
  num_rows      1
  num_columns    25
  variable       entlong, 0,11
  leader         "Longitude:"
  helpfile       help/markmap.hlp
}

```

```

DATUM entlat {
  num_rows      1
  num_columns    25
  variable       entlat, 0,11
  leader         "Latitude: "
  helpfile       help/markmap.hlp
}

```

```

-----
-      Declarations for show coordinates
-----

```

```

-----
DATUM shwlat {
    num_rows 1
    num_columns 14
    variable shwlat, 0,0
    pickable NO
    helpfile help/contmap.hlp
}

DATUM shwlong {
    num_rows 1
    num_columns 14
    variable shwlong, 0,0
    pickable NO
    helpfile help/contmap.hlp
}

DATUM prefazdist {
    num_rows 1
    num_columns 8
    variable prefazdist, 0,4
    pickable NO
}

DATUM curfindist {
    num_rows 1
    num_columns 26
    variable curfindist, 0,16
    leader "Fix distance: "
    helpfile help/curfindi.hlp
}

DATUM prefazrad {
    num_rows 1
    num_columns 8
    variable prefazrad, 0,4
    pickable NO
}

DATUM curfixrad {
    num_rows 1
    num_columns 26
    variable curfixrad, 0,16
    leader "Fix radial: "
    helpfile help/curfixra.hlp
}

DATUM prefazid {
    num_rows 1
    num_columns 10
    variable prefazid, 0,4
    pickable NO
}

DATUM curfixid {
    num_rows 1
    num_columns 26
    variable curfixid, 0,16
    leader "Fix ID: "
    helpfile help/curfixid.hlp
}

DATUM prenavpt {
    num_rows 1
    num_columns 10
    variable prenavpt, 0,4
    pickable NO
}

DATUM curnavpt {
    num_rows 1
    num_columns 24
    variable curnavpt, 0,16
    leader "Nav. Point: "
    helpfile help/curnavpt.hlp
}

DATUM newstrms {
    num_rows 1
    num_columns 40
    variable newstrms, 0,7
}

```

```

leader      "Name:"
helpfile    help/atmstrm.hlp
}

WINDOW nextnavpt (
  num_rows 1
  num_columns 35
  textline  nextnavptxt, 0, 2
  helpfile  help/nextnav.hlp
)

-----

TEXTLINE retmtrtext ("Continue without selecting MTR")
TEXTLINE retmtrtext1 ("Continue without creating mission")
TEXTLINE retmtrtext2 ("Save mission in database")
TEXTLINE retmtrmis ("Continue without selecting mission")

WINDOW retmtrtext (
  num_rows 1
  num_columns 40
  textline  retmtrtext, 0, 2
  helpfile  help/mcomstr.hlp
)

WINDOW retmtrtext1 (
  num_rows 1
  num_columns 40
  textline  retmtrtext1, 0, 2
  helpfile  help/mcomiss.hlp
)

WINDOW retmtrtext2 (
  num_rows 1
  num_columns 30
  textline  retmtrtext2, 0, 2
  helpfile  help/savemiss.hlp
)

WINDOW retmtrmis (
  num_rows 1
  num_columns 40
  textline  retmtrmis, 0, 2
  helpfile  help/mcomemis.hlp
)

-----

WINDOW defmcomtr (
  num_rows 15
  num_columns 78
  textline  curpretxt, 0, 16
  textline  curpretxt3, 1, 16
  datum     curnavpt, 2, 1, "CALL VCAPITAL &curnavpt",
  "NEWVALS"
  datum     prenavpt, 2, 26
  datum     curfixid, 3, 1, "CALL VCAPITAL &curfixid",
  "NEWVALS"
  datum     prefixid, 3, 26
  datum     curfixrad, 4, 1
  datum     prefixrad, 4, 26
  datum     curfindist, 5, 1
  datum     prefixdist, 5, 26
  datum     entlat, 2, 38, "CALL lat2dec &ent",
  "NEWVALS"
  datum     shwlat, 2, 64
  datum     entlong, 3, 38, "CALL lon2dec &ent",
  "NEWVALS"
  datum     shwlong, 3, 64
  datum     curfixtype, 4, 38, "CALL VCAPITAL &curfixtype",
  "NEWVALS"
  datum     prefixtype, 4, 64
  datum     curlowalt, 7, 2, "CALL alt2dec &curlowalt",
  "NEWVALS"
  datum     prelowalt, 7, 26
  datum     curhighalt, 8, 2, "CALL alt2dec &curhighalt",
  "NEWVALS"
  datum     prehightalt, 8, 26
  datum     curwidleft, 9, 2
  datum     prewidleft, 9, 27
  datum     curwidright, 10, 2
  datum     prewidright, 10, 27

```



```

num_rows 1
num_columns 40
textline definestrst, 0, 2
helpfile help/getmap.hlp
)

WINDOW definestr {
    num_rows 9
    num_columns 77
    datum      origintr, 0, 1
    datum      schedule, 2, 1
    datum      roads, 4, 1
    datum      datapubl, 6, 1
    button      definestr, 8, 1,      "CALL eststrpt"
}

SCREEN mtrdefinescreen {
    title      "MTR DEFINITION"
    window      mtrsmocn, 3, 1
    window      definestr, 7, 2
    window      mtraction, 20, 1
    border      YES
}

-----
-               Declarations for MISSION REQUIREMENTS WINDOW
-----

VARIABLE ac_in_form {
    type      INTEGER
    format    %2d
    lowlimit  1
    uplimit   16
    default   2
}

VARIABLE prstunt {
    foundia    "VARCHAR pr_pwr_u.arn"
    type      STRING
    format     %s
}

DATUM prstunt {
    num_rows 1
    num_columns 10
    variable  prstunt, 0, 2
    leader    "["
    trailer   "]"
    pickable  NO
}

VARIABLE mistype {
    type      STRING
    format     %s
}

DATUM mistype1 {
    num_rows 1
    num_columns 10
    variable  mistype, 0, 14
    leader    "Mission type:"
    helpfile  help/mistype.hlp
}

DATUM mistype {
    num_rows 1
    num_columns 10
    variable  mistype, 0, 14
    leader    "Mission type:"
    pickable  NO
}

VARIABLE prealtlev {
    foundia    "ALTSPEC prealtlev.spec"
    type      STRING
    format     %s
}

DATUM prealtlev {
    num_rows 1
    num_columns 10
    variable  prealtlev, 0, 0
}

```

```

        picktable      NO
    }

    VARIABLE curaltlev {
        foundin      "ALTSPMC curlowalt.spec"
        type         STRING
        format       %s
    }

    DATUM curaltlev {
        num_rows      1
        num_columns    21
        variable       curaltlev, 0, 11
        leader         "Alt: "
        helpfile       help/altmtr.hlp
    }

    VARIABLE ac_pre_pwr {
        type          DOUBLE
        format        %12.3lf
        lowlimit       0.0
        uplimit        4000.0
        default        100.0
    }

    DATUM prapreset {
        num_rows      1
        num_columns    12
        variable       ac_pre_pwr, 0, 0
        picktable      NO
    }

    VARIABLE ac_cur_pwr {
        type          DOUBLE
        format        %10.3lf
        lowlimit       0.0
        uplimit        4000.0
        default        100.0
    }

    DATUM curpreset {
        num_rows      1
        num_columns    21
        variable       ac_cur_pwr, 0, 11
        leader         "Power: "
        helpfile       help/pwrspc.hlp
    }

    VARIABLE ac_pre_spd {
        type          INTEGER
        format        %3d
    }

    DATUM prespeed {
        num_rows      1
        num_columns    9
        variable       ac_pre_spd, 0, 0
        trailer        "KTS"
        picktable      NO
    }

    VARIABLE ac_cur_spd {
        type          INTEGER
        format        %3d
        lowlimit       100
        uplimit        600
        default        450
    }

    DATUM curspeed {
        num_rows      1
        num_columns    21
        variable       ac_cur_spd, 0, 11
        leader         "Speed: "
        trailer        "KTS"
        helpfile       help/mtrspeed.hlp
    }

    DATUM preasvpt1 {
        num_rows      1
        num_columns    3
        variable       preasvpt, 0, 0
    }

```

```

        pickable      NO
    )

    DATUM curnavpt1 {
        num_rows      1
        num_columns    21
        variable       curnavpt, 0, 11
        leader         "Waypoint:"
        helpfile       help/navname.hlp
    }

    TEXTLINE cancelparatxt ("Cancel data entry for this mission")

    BUTTON cancelpara {
        num_rows      1
        num_columns    35
        textline       cancelparatxt, 0, 2
        helpfile       help/cancelstr.hlp
    }

    TEXTLINE saveparatxt ("Save this mission's data")

    BUTTON savepara {
        num_rows      1
        num_columns    30
        textline       saveparatxt, 0, 2
        helpfile       help/savemiss.hlp
    }

    TEXTLINE operdatatxt ("Ready to enter operational data")

    BUTTON operdata {
        num_rows      1
        num_columns    33
        textline       operdatatxt, 0, 2
        helpfile       help/entropc.hlp
    }

    TEXTBLOCK mtr {
        num_rows      10
        num_columns    38
        filename       txtblk/mtr.txt
    }

    TEXTLINE mtrlabeltxt ("NAV    FIX    FIX TYPE    WIDTH")

    TEXTLINE mtrtxt ("REFERENCE INFORMATION FOR DATA ENTRY")

    TEXTLINE curpretxt1 ("CURRENT    PREVIOUS")

    TEXTLINE curpretxt2 ("-----    -----")

    WINDOW opentry {
        num_rows      10
        num_columns    78
        button opetandy, 2, 19, "CALL psaddops 1"
        button opesasea, 7, 19, "CALL psaddops 12"
    }

    WINDOW fltpara {
        num_rows      17
        num_columns    78
        datum          mistype, 0, 2
        datum          acmsme, 1, 2
        textline        mtrtxt, 0, 40
        textline        mtrlabeltxt, 2, 38
        textline        curpretxt1, 3, 14
        textline        curpretxt2, 4, 14
        datum          curnavpt1, 5, 1, "CALL VCAPITAL 6curnavpt",
                                "NEWVALS"
        datum          presnavpt1, 5, 25
        datum          curspeed, 7, 1
        datum          prespeed, 7, 25
        datum          curprsrct, 9, 2
        datum          preprrsct, 9, 22
        datum          prsrctunt, 10, 15
        datum          curaltlev, 12, 2, "CALL alt2dec 6curlowalt",
                                "NEWVALS"
        datum          prealtlev, 12, 23
        box            1, 37, 14, 77
        line           3, 38, 3, 78
        line           15, 0, 15, 77
    }

```

```

        button      nextnavpt, 14, 2, "CALL nextnavpt"
        button      opardata, 16, 2, "NEW_SCREEN opentry"
        button      cancelpara, 16,40, "CALL cancel"
        textblock   mtr,      4,38
    }

SCREEN mtrflt {
    title           "FLIGHT PARAMETER ENTRY"
    window          mtrmnom,   2,1
    window          fltpara,    5,1
    border          YES
}

SCREEN opentry {
    title           "FLIGHT OPERATION DATA ENTRY FOR MTR"
    window          mtrmnom,    3,1
    window          opentry,    6,1
    border          YES
}

```

Declarations for seasonal MTR and REQUIREMENTS

```

DATUM decnite {
    num_rows 1
    num_columns 6
    variable  decnite, 0, 0
    helpfile  help/mnthnrite.hlp
}

DATUM decday {
    num_rows 1
    num_columns 10
    variable  decday, 0, 5
    leader    "DEC:"
    helpfile  help/mnthday.hlp
}

DATUM novnite {
    num_rows 1
    num_columns 6
    variable  novnite, 0, 0
    helpfile  help/mnthnrite.hlp
}

DATUM novday {
    num_rows 1
    num_columns 10
    variable  novday, 0, 5
    leader    "NOV:"
    helpfile  help/mnthday.hlp
}

DATUM octnite {
    num_rows 1
    num_columns 6
    variable  octnite, 0, 0
    helpfile  help/mnthnrite.hlp
}

DATUM octday {
    num_rows 1
    num_columns 10
    variable  octday, 0, 5
    leader    "OCT:"
    helpfile  help/mnthday.hlp
}

DATUM sepnite {
    num_rows 1
    num_columns 6
    variable  sepnite, 0, 0
    helpfile  help/mnthnrite.hlp
}

DATUM sepday {
    num_rows 1
    num_columns 10
    variable  sepday, 0, 5
    leader    "SEP:"
    helpfile  help/mnthday.hlp
}

```

```

DATUM augnrite {
    num_rows 1
    num_columns 6
    variable augnrite, 0, 0
    helpfile help/mnthnrite.hlp
}

DATUM augday {
    num_rows 1
    num_columns 10
    variable augday, 0, 5
    loader "AUG:"
    helpfile help/mnthday.hlp
}

DATUM julaite {
    num_rows 1
    num_columns 6
    variable julaite, 0, 0
    helpfile help/mnthnrite.hlp
}

DATUM julday {
    num_rows 1
    num_columns 10
    variable julday, 0, 5
    loader "JUL:"
    helpfile help/mnthday.hlp
}

DATUM junnrite {
    num_rows 1
    num_columns 6
    variable junnrite, 0, 0
    helpfile help/mnthnrite.hlp
}

DATUM juneday {
    num_rows 1
    num_columns 10
    variable juneday, 0, 5
    loader "JUN:"
    helpfile help/mnthday.hlp
}

DATUM maynrite {
    num_rows 1
    num_columns 6
    variable maynrite, 0, 0
    helpfile help/mnthnrite.hlp
}

DATUM mayday {
    num_rows 1
    num_columns 10
    variable mayday, 0, 5
    loader "MAY:"
    helpfile help/mnthday.hlp
}

DATUM aprnrite {
    num_rows 1
    num_columns 6
    variable aprnrite, 0, 0
    helpfile help/mnthnrite.hlp
}

DATUM aprday {
    num_rows 1
    num_columns 10
    variable aprday, 0, 5
    loader "APR:"
    helpfile help/mnthday.hlp
}

DATUM marnrite {
    num_rows 1
    num_columns 6
    variable marnrite, 0, 0
    helpfile help/mnthnrite.hlp
}

DATUM marday {

```

```

    num_rows 1
    num_columns 10
    variable marday, 0, 5
    leader "MAR:"
    helpfile help/mnthday.hlp
}

DATUM febaite {
    num_rows 1
    num_columns 6
    variable febaite, 0, 0
    helpfile help/mnthaito.hlp
}

DATUM febday {
    num_rows 1
    num_columns 10
    variable febday, 0, 5
    leader "FEB:"
    helpfile help/mnthday.hlp
}

DATUM janaite {
    num_rows 1
    num_columns 6
    variable janaite, 0, 0
    helpfile help/mnthaito.hlp
}

DATUM janday {
    num_rows 1
    num_columns 10
    variable janday, 0, 5
    leader "JAN:"
    helpfile help/mnthday.hlp
}

WINDOW month {
    num_rows 14
    num_columns 77
    textline instritxt, 1, 1
    textline dayaito, 3, 0
    datum janday, 4, 1
    datum janaite, 4, 12
    datum febday, 5, 1
    datum febaite, 5, 12
    datum marday, 6, 1
    datum maraite, 6, 12
    datum aprday, 4, 20
    datum apraite, 4, 31
    datum mayday, 5, 20
    datum mayaito, 5, 31
    datum junday, 6, 20
    datum junaite, 6, 31
    datum julday, 4, 30
    datum julaite, 4, 30
    datum augday, 5, 30
    datum augaite, 5, 30
    datum sepday, 6, 30
    datum sepaite, 6, 30
    datum octday, 4, 30
    datum octaite, 4, 30
    datum novday, 5, 30
    datum novaite, 5, 30
    datum decday, 6, 30
    datum decbite, 6, 30
    button savepara, 13, 2, "CALL savamis"
    button canclpara, 13, 40, "CALL canclmis"
}

```

- Declarations for new SELECT AIRCRAFT AND MISSION FOR MFR

TEXTLINE shumorenistxt ("Show more mission names (if any)")

```

BUTTON shumorenis {
    num_rows 1
    num_columns 38
    textline shumorenistxt, 0, 2
    helpfile help/shumorenis.hlp
}

```

```

TEXTLINE recallmis ("Recall one of the following missions")

WINDOW newmiss {
    num_rows      1
    num_columns   40
    datum         misname, 0, 2
}

WINDOW curmiss {
    num_rows      16
    num_columns   78
    button staemiss, 1, 2, "ADD WINDOW newmiss 7 3",
                        "UPDATE DATUM misname",
                        "CALL psuamiss aid.arr"
    button shumoremis, 1, 40, "CALL sabench"
    textline recallmis, 3, 2
    datum maldat00, 4, 4
    datum maldat01, 5, 4
    datum maldat02, 6, 4
    datum maldat03, 7, 4
    datum maldat04, 8, 4
    datum maldat05, 9, 4
    datum maldat06, 10, 4
    datum maldat07, 11, 4
    datum maldat08, 12, 4
    datum maldat09, 13, 4
    datum maldat10, 4, 42
    datum maldat11, 5, 42
    datum maldat12, 6, 42
    datum maldat13, 7, 42
    datum maldat14, 8, 42
    datum maldat15, 9, 42
    datum maldat16, 10, 42
    datum maldat17, 11, 42
    datum maldat18, 12, 42
    datum maldat19, 13, 42
    button malbut00, 4, 2, "CALL MISsconn deplmult[0].arr"
    button malbut01, 5, 2, "CALL MISsconn deplmult[1].arr"
    button malbut02, 6, 2, "CALL MISsconn deplmult[2].arr"
    button malbut03, 7, 2, "CALL MISsconn deplmult[3].arr"
    button malbut04, 8, 2, "CALL MISsconn deplmult[4].arr"
    button malbut05, 9, 2, "CALL MISsconn deplmult[5].arr"
    button malbut06, 10, 2, "CALL MISsconn deplmult[6].arr"
    button malbut07, 11, 2, "CALL MISsconn deplmult[7].arr"
    button malbut08, 12, 2, "CALL MISsconn deplmult[8].arr"
    button malbut09, 13, 2, "CALL MISsconn deplmult[9].arr"
    button malbut10, 4, 40, "CALL MISsconn deplmult[10].arr"
    button malbut11, 5, 40, "CALL MISsconn deplmult[11].arr"
    button malbut12, 6, 40, "CALL MISsconn deplmult[12].arr"
    button malbut13, 7, 40, "CALL MISsconn deplmult[13].arr"
    button malbut14, 8, 40, "CALL MISsconn deplmult[14].arr"
    button malbut15, 9, 40, "CALL MISsconn deplmult[15].arr"
    button malbut16, 10, 40, "CALL MISsconn deplmult[16].arr"
    button malbut17, 11, 40, "CALL MISsconn deplmult[17].arr"
    button malbut18, 12, 40, "CALL MISsconn deplmult[18].arr"
    button malbut19, 13, 40, "CALL MISsconn deplmult[19].arr"
    button retacmis, 15, 2, "CALL seliste",
                        "CALL NEW_SCREEN oldscreen"
}

TEXTLINE selmisstxt ("Select mission")

BUTTON selmiss {
    num_rows      1
    num_columns   38
    textline      selmisstxt, 0, 2
    helpfile      help/selmiss.hlp
}

TEXTLINE fltparinfst ("Specify flight parameter information")

BUTTON fltparinf {
    num_rows      1
    num_columns   37
    textline      fltparinfst, 0, 2
    helpfile      help/fltparam.hlp
}

WINDOW selacmiss {
    num_rows      15
    num_columns   77
    datum acnamel, 1, 1, "CALL vflyacstr tid.arr"
}

```



```

        button selmiss, 3,2, "CALL pchgmis"
        button fltparinf, 5,2, "CALL pcutrflt"
        button retatrent1, 13,2, "CALL cancmis"
    }

    SCREEN spctrmis {
        title "SELECT AIRCRAFT AND MISSION FOR MFR"
        window stramocml, 3,1
        window selamiss, 7,1
        border YES
    }

    SCREEN chgmis {
        title "SELECT AIRCRAFT AND MISSION FOR MFR"
        window stramocml, 2,1
        window cumiss, 6,1
        border YES
    }

-----
-      Declarations for Day-Night window
-----

    DATUM night {
        num_rows 1
        num_columns 16
        variable night, 0, 8
        leader "Night:"
        helpfile help/niteops.hlp
    }

    DATUM day {
        num_rows 1
        num_columns 16
        variable day, 0, 6
        leader "Day:"
        helpfile help/dayops.hlp
    }

    WINDOW daynite {
        num_rows 12
        num_columns 77
        datum scame, 1, 1
        textline instrtxt, 3, 1
        datum day, 5, 2
        datum night, 8,13
        button savepara, 10, 2, "CALL expmops",
                                "CALL savemis"
        button cancelpara, 10,40, "CALL cancmis"
        border YES
    }

-----
-      Declarations for mission specification window
-----

    DATUM sortie {
        num_rows 1
        num_columns 37
        variable sq_in_form, 0, 35
        leader "Number of aircraft in formation: "
        helpfile help/sunform.hlp
    }

    VARIABLE misdesc {
        foundin "VARCHAR misdesc.arr"      -this description is for mission
        type STRING
        format *-60s
    }

    DATUM misdesc1 {
        num_rows 1
        num_columns 76
        variable misdesc, 0, 7
        leader "Descr:"
        pickable NO
    }

    DATUM misdesc {
        num_rows 1
        num_columns 76
        variable misdesc, 0, 14
    }

```

```

        leader      "Description:"
        helpfile    help/misdesc.hlp
    )

WINDOW mstramocm3 {
    num_rows      3
    num_columns   78
    line          2,0, 2,77
    datum         misamcm1, 0, 2
    datum         misdesc1, 1, 2
}

WINDOW misspec {
    num_rows      16
    num_columns   77
    datum         misdesc, 1,1
    datum         mistypel, 3,2
    datum         sortia, 5,2
    button         retxtrest1, 14,2, "CALL NEW_SCREEN oldscreen"
    button         retxtrest2, 14,47, "CALL external"
}

SCREEN misspec {
    title         "MISSION SPECIFICATION FOR AN MTR"
    window        mstramocm3, 3,1
    window        misspec, 6,1
    border        YES
}

-01-14-88 st  will try to fix this file so it will be only for assessments
-01-14-88 st  changes will be commented with date
-01-16-88 st  change title on first screen, change lat. long. format loc
-
-01-17-88 st  can't to fix sdf file commented with 1-17-88

-----
- DECLARATIONS FOR SCREEN HEADER
-----

VARIABLE assemame {
    foundin      "ASAMHEADR ASSESSMENT.name"
    type         STRING
    format       %30s
}

DATUM assemame {
    num_rows     1
    num_columns  50
    variable     assemame, 0, 18
    leader       "Assessment name: "
    pickable     NO
}

DATUM assemame1 {
    num_rows     1
    num_columns  75
    variable     assemame, 0, 30
    leader       "Name of current assessment: "
    pickable     NO
}

VARIABLE comment {
    foundin      "ASAMHEADR ASSESSMENT.desc"
    type         STRING
    format       %66s
}

DATUM comment {
    num_rows     1
    num_columns  76
    variable     comment, 0, 10
    leader       "Comment: "
    pickable     NO
}

VARIABLE curmapnum {
    type         STRING
    format       %30s
}

DATUM curmapnum {
    num_rows     1

```

```

        num_columns 77
        variable     curmapam, 0, 19
        loader       "Current map name: "
        pickable     NO
    }

WINDOW lis {
    num_rows         1
    num_columns      78
    line             0, 0, 0, 77
}

WINDOW asseamcom {
    num_rows         3
    num_columns      78
    datum            asseamcom1, 0, 1
    datum            comment, 1, 2
    line             2, 0, 2, 77
}

WINDOW asseamcom1 {
    num_rows         4
    num_columns      78
    datum            asseamcom1, 0, 0
    datum            comment, 1, 1
    datum            curmapam, 2, 1
    line             3, 0, 3, 77
}

-----
-   Declarations for buttons used in majoraction footer
-----

TEXTLINE probstat1 { "REVIEW CURRENT ASSESSMENT STATUS" }

BUTTON probstat {
    num_rows         1
    num_columns      35
    textline         probstat1, 0, 2
    helpfile         help/probstat.hlp
}

TEXTLINE probstat2 { "REVIEW ASSESSMENT STATUS" }

BUTTON probstat2 {
    num_rows         1
    num_columns      35
    textline         probstat2, 0, 2
    helpfile         help/revstat.hlp
}

TEXTLINE probdef { "ADD INFORMATION TO CURRENT ASSESSMENT" }

BUTTON probdef {
    num_rows         1
    num_columns      39
    textline         probdef, 0, 2
    helpfile         help/probdef.hlp
}

TEXTLINE probdef1 { "ADD TO ASSESSMENT DEFINITION" }

BUTTON probdef1 {
    num_rows         1
    num_columns      31
    textline         probdef1, 0, 2
    helpfile         help/probdef.hlp
}

TEXTLINE analysis { "ANALYSE DATA" }

BUTTON analysis {
    num_rows         1
    num_columns      15
    textline         analysis, 0, 2
    helpfile         help/datanal.hlp
}

TEXTLINE reportgen { "MAKE A REPORT" }

BUTTON reportgen {
    num_rows         1

```

```

num_columns 16
textline    reportgen, 0, 2
helpfile    help/reportgen.hlp
}

TEXTLINE vechklistst { "VIEW CHECKLIST FOR CURRENT ASSESSMENT" }

BUTTON vechklist {
  num_rows    1
  num_columns 42
  textline    vechklistst, 0, 2
  helpfile    help/viewlist.hlp
}

-----
-      Declarations for majoraction footers
-----

WINDOW majoraction {      -header for problem status screen
  num_rows    4
  num_columns 78
  line        0, 0, 0.77
  title       "Alternative actions you can now take:"
  button      probdef,    2, 2, "NEW_SCREEN probdefscreen"
  button      analysis,   2, 56, "NEW_SCREEN analysiscreen"
  button      vechklist,  3, 2, "NEW_SCREEN viewchecklistscreens"
  button      reportgen,  3, 56, "CALL dummy2"
}

WINDOW majoraction1 {      -header for problem definition screen
  num_rows    4
  num_columns 78
  line        0, 0, 0.77
  title       "Alternative actions you can now take:"
  button      analysis,   2, 2, "NEW_SCREEN analysiscreen"
  button      probstat2,  2, 46, "CALL pprobstat"
  button      vechklist,  3, 2, "NEW_SCREEN viewchecklistscreens"
  button      reportgen,  3, 46, "CALL dummy2"
}

WINDOW majoraction2 {      -header for data analysis screen
  num_rows    4
  num_columns 78
  line        0, 0, 0.77
  title       "Alternative actions you can now take:"
  button      probdef1,   2, 2, "NEW_SCREEN probdefscreen"
  button      reportgen,  2, 47, "CALL dummy2"
  button      vechklist,  3, 2, "NEW_SCREEN viewchecklistscreens"
  button      probstat2,  3, 47, "CALL pprobstat"
}

WINDOW majoraction3 {      -header for report generation screen
  num_rows    4
  num_columns 78
  line        0, 0, 0.77
  title       "Alternative actions you can now take:"
  button      vechklist,  2, 2, "NEW_SCREEN viewchecklistscreens"
  button      probstat2,  2, 47, "CALL pprobstat"
  button      probdef1,   3, 2, "NEW_SCREEN probdefscreen"
  button      analysis,   3, 47, "NEW_SCREEN analysiscreen"
}

WINDOW majoraction4 {      -header for view checklist screen
  num_rows    4
  num_columns 78
  line        0, 0, 0.77
  title       "Alternative actions you can now take:"
  button      probstat2,  2, 2, "CALL pprobstat"
  button      reportgen,  2, 47, "CALL dummy2"
  button      probdef1,   3, 2, "NEW_SCREEN probdefscreen"
  button      analysis,   3, 47, "NEW_SCREEN analysiscreen"
}

TEXTLINE loadglobal {"LOAD LOCAL DATA FOR GENERAL ACCESS"}

BUTTON loadglobal {
  num_rows    1
  num_columns 39
  textline    loadglobal, 0, 2
  helpfile    help/loadglob.hlp
}

```

```

WINDOW majoraction3 {      -footer for database housekeeping screen
    num_rows      4
    num_columns    78
    line          0, 0, 0,77
    title         "Alternative actions you can now take:"
    button        loadglobal, 2, 2, "CALL dummy"
    button        assessment, 3, 2, "CALL peprobet"
}

SCREEN mstrtry {
    title         "MFR DATA ENTRY"
    window        mstrscreen, 3,1
    window        mstrdataentry, 7,1
    window        majoraction3, 18,1
    border        YES
}

-----
-               Declarations for problem status screen
-----

VARIABLE startdate {
    foundin       "VARCHAR startdate.arr"
    type          STRING
    format        4-21s
}

DATUM startdate {
    num_rows      1
    num_columns    40
    variable       startdate, 0,15
    leader         "Date started: "
    pickable       NO
}

VARIABLE lastdate {
    foundin       "VARCHAR lastdate.arr"
    type          STRING
    format        4-21s
}

DATUM lastdate {
    num_rows      1
    num_columns    55
    variable       lastdate, 0, 28
    leader         "Date of last modification: "
    pickable       NO
}

VARIABLE planrlast {
    foundin       "VARCHAR planrlast.arr"
    type          STRING
    format        4-30s
}

DATUM planrlast {
    num_rows      1
    num_columns    55
    variable       planrlast, 0,19
    leader         "Last modified by: "
    pickable       NO
}

WINDOW probetat {
    num_rows      5
    num_columns    78
    datum         startdate, 0, 2
    datum         lastdate, 2, 2
    datum         planrlast, 4, 2
}

TEXTLINE chgasstxt ("Work on a different assessment")

BOTTOM chgas {
    num_rows      1
    num_columns    70
    textline       chgasstxt, 0, 2
    helpfile       help/probdaf.hlp
}

WINDOW chgasess {
    -change to different assessment

```

```

    sum_rows 1
    sum_columns 76
    button chgass, 0, 2, "CALL pschgass"
}

-----
-      Declaration for ENVIRONMENTAL ASSESSMENT DEFINITION SCREEN
-----

TEXTLINE modmoetxt {"Work with MDA information (number or type of aircraft, missions, etc.)"}

BUTTON modmoec {
    sum_rows 1
    sum_columns 76
    textline modmoetxt, 0, 2
    helpfile help/moework.hlp
}

TEXTLINE modstrtxt {"Work with MFR information (number or type of aircraft, missions, etc.)"}

BUTTON modstr {
    sum_rows 1
    sum_columns 76
    textline modstrtxt, 0, 2
    helpfile help/strwork.hlp
}

TEXTLINE modmaptxt {"Work with map information (designate land uses, update maps)"}

BUTTON modmap {
    sum_rows 1
    sum_columns 76
    textline modmaptxt, 0, 2
    helpfile help/mapwork.hlp
}

TEXTLINE selectxt {"Actions you can now take to add information to this assessment:"}

WINDOW selection {
    sum_rows 7
    sum_columns 77
    textline selectxt, 0, 1
    button modmap, 2, 1, "CALL dummy2"
    button modstr, 4, 1, "CALL psentreat"
    button modmoec, 6, 1, "CALL dummy"
}

SCREEN probdefscreen {
    title "ENVIRONMENTAL ASSESSMENT DEFINITION"
    window assnmocm, 3, 1
    window selection, 7, 2
    window majoraction1, 18, 1
    border YES
}

-----
-      Declaration for entering coordinates window
-----

TEXTLINE entupplf
{"Enter upper-left corner coordinates of area of current interest"}

TEXTLINE entlowrt
{"Enter lower right corner coordinates of area of current interest"}

DATUM shwlat2 {
    sum_rows 1
    sum_columns 25
    variable shwlat, 0, 11
    leader "Latitude: "
    helpfile help/ocahmap.hlp
}

DATUM shwlong2 {
    sum_rows 1
    sum_columns 25
    variable shwlong, 0, 12
    leader "Longitude: "
    helpfile help/ocahmap.hlp
}

WINDOW entocor {

```

```

num_rows      10
num_columns   78
textline      extupplf, 2, 2
datum         extlat, 3, 1, "CALL lat2dec &ext", "NEWVALS"
datum         extlong, 3, 38, "CALL lon2dec &ext", "NEWVALS"
textline      extlowrt, 5, 2
datum         shwlat2, 6, 1, "CALL lat2dec &show", "NEWVALS"
datum         shwlong2, 6, 38, "CALL lon2dec &show", "NEWVALS"
)

-----
~      Declarations for DATA ANALYSIS SCREEN
-----

TEXTLINE geodatingtxt ("Make geodata inquiries on map screen")

BUTTON geodating {
  num_rows      1
  num_columns   78
  textline      geodatingtxt, 0, 2
  helpfile      help/geodating.hlp
}

TEXTLINE compnoisefxt ("Compare noise effects") -1-17-88

BUTTON compnoisef {
  num_rows      1
  num_columns   76
  textline      compnoisefxt, 0, 2
  helpfile      help/compnois.hlp
} -1-17-88 -2-06-88

TEXTLINE calcmnoisefxt ("Calculate noise effects in specified area")

BUTTON calcmnoisef {
  num_rows      1
  num_columns   76
  textline      calcmnoisefxt, 0, 2
  helpfile      help/afcalc.hlp
}

TEXTLINE calcmnoisexpxt ("Calculate noise exposure in specified area")

BUTTON calcmnoisexp {
  num_rows      1
  num_columns   76
  textline      calcmnoisexpxt, 0, 2
  helpfile      help/expcalc.hlp
}

TEXTLINE calclqlooktxt ("Calculate quicklook (point) exposure estimate")

BUTTON calclqlook {
  num_rows      1
  num_columns   50
  textline      calclqlooktxt, 0, 2
  helpfile      help/qlklook.hlp
}

TEXTLINE entcoortxt ("Enter coordinates from keyboard")

BUTTON entcoor {
  num_rows      1
  num_columns   40
  textline      entcoortxt, 0, 2
  helpfile      help/kbentry.hlp
}

TEXTLINE usemaptxt ("Use map screen")

BUTTON usemap {
  num_rows      1
  num_columns   30
  textline      usemaptxt, 0, 2
  helpfile      help/usemap.hlp
}

TEXTLINE selectanactxt ("Actions you can now take to analyze environmental assessment data:")

TEXTLINE defgeoareactxt ("Specify a geographic area of interest:")

WINDOW datanaction {
  num_rows      11

```

```

num_columns 78
textline selectasstxt, 0, 2
button calogklook, 2, 2, "CALL dummy"
button calmoisexp, 3, 2, "CALL dummy"
button calmoisaf, 4, 2, "CALL dummy"
button compoisaf, 5, 2, "CALL dummy"
button geodating, 6, 2, "CALL dummy"
textline dafgeocrestxt, 8, 2
button usemap, 9, 2, "CALL dummy2"
button antacox, 10, 2, "ADD_WINDOW antacox 5 1"
}

SCREEN analysiscreen {
title "DATA ANALYSIS"
window assnamoon, 2, 1
window dataaction, 6, 1
window majoraction2, 18, 1
border THS
}

TEXTLINE showmoretxt ("Show more assessments (if any)")

BUTTON showmore {
num_rows 1
num_columns 37
textline showmoretxt, 0, 2
helpfile help/getassmnt.hlp
}

TEXTLINE recallsstxt ("Recall one of the following assessments:")

TEXTLINE stnewasstxt ("Start a new assessment")

BUTTON stnewass {
num_rows 1
num_columns 28
textline stnewasstxt, 0, 2
helpfile help/newassmnt.hlp
}

VARIABLE newassnam {
foundia "VARCHAR a2bv.arr"
type STRING
format 4-30s
}

DATUM newassnam {
num_rows 1
num_columns 40
variable newassnam, 0, 7
loader "Name:"
helpfile help/stnewass.hlp
}

WINDOW newassnam {
num_rows 1
num_columns 40
datum newassnam, 0, 1
}

WINDOW stnewass {
num_rows 1
num_columns 78
button stnewass, 0, 2, "ADD_WINDOW newassnam, 6 3",
"UPDATE_DATUM newassnam",
"CALL penwasm a2bv.arr"
button showmore, 0, 41, "CALL ubunch"
}

WINDOW recalls {
num_rows 1
num_columns 78
textline recallsstxt, 0, 2
}

WINDOW curassbut {
num_rows 10
num_columns 2
button mulbut00, 0, 0, "CALL ASANwcon default[0].arr"
button mulbut01, 1, 0, "CALL ASANwcon default[1].arr"
button mulbut02, 2, 0, "CALL ASANwcon default[2].arr"
button mulbut03, 3, 0, "CALL ASANwcon default[3].arr"
}

```



```

        button      subbut04, 4, 0, "CALL ASANWocaa deplmult[4].arr"
        button      subbut05, 5, 0, "CALL ASANWocaa deplmult[5].arr"
        button      subbut06, 6, 0, "CALL ASANWocaa deplmult[6].arr"
        button      subbut07, 7, 0, "CALL ASANWocaa deplmult[7].arr"
        button      subbut08, 8, 0, "CALL ASANWocaa deplmult[8].arr"
        button      subbut09, 9, 0, "CALL ASANWocaa deplmult[9].arr"
    }

WINDOW curasabut1 {
    num_rows      10
    num_columns    2
    button      subbut10, 0, 0, "CALL ASANWocaa deplmult[10].arr"
    button      subbut11, 1, 0, "CALL ASANWocaa deplmult[11].arr"
    button      subbut12, 2, 0, "CALL ASANWocaa deplmult[12].arr"
    button      subbut13, 3, 0, "CALL ASANWocaa deplmult[13].arr"
    button      subbut14, 4, 0, "CALL ASANWocaa deplmult[14].arr"
    button      subbut15, 5, 0, "CALL ASANWocaa deplmult[15].arr"
    button      subbut16, 6, 0, "CALL ASANWocaa deplmult[16].arr"
    button      subbut17, 7, 0, "CALL ASANWocaa deplmult[17].arr"
    button      subbut18, 8, 0, "CALL ASANWocaa deplmult[18].arr"
    button      subbut19, 9, 0, "CALL ASANWocaa deplmult[19].arr"
}

TEXTLINE retacess ("Continue without selecting assessment")

BUTTON retasess {
    num_rows      1
    num_columns    40
    textline      retacess, 0, 2
    helpfile      help/retasess.hlp
}

WINDOW retwocaa {
    num_rows      1
    num_columns    76
    button      retasess, 0, 1, "CALL alistc", "CALL peprobet"
}

SCREEN abgcurascreen {
    title          "SELECT ANOTHER ASSESSMENT"
    window         asanwocaa, 2, 1
    window         stacwocaa, 6, 1
    window         recalass, 8, 1
    window         curasabut, 9, 3
    window         curasabut1, 9, 42
    window         curdet, 9, 5
    window         curdet1, 9, 44
    window         retwocaa, 20, 2
    border         YES
}

-----
~      Declarations for NEW ASSESSMENT DEFINITION
-----

VARIABLE entdesc1 {
    foundin      "VARCHAR60 entdesc[0].arr"
    type         STRING
    format       4-60s
}

DATUM entdesc1 {
    num_rows      1
    num_columns    75
    variable      entdesc1, 0, 0
    helpfile      help/entdesc.hlp
}

VARIABLE entdesc2 {
    foundin      "VARCHAR60 entdesc[1].arr"
    type         STRING
    format       4-60s
}

DATUM entdesc2 {
    num_rows      1
    num_columns    75
    variable      entdesc2, 0, 0
    helpfile      help/entdesc.hlp
}

VARIABLE entdesc3 {

```

```

        foundia    "VARCHAR60 entdesc[2].arr"
        type       STRING
        format     4-60s
    }

    DATUM    entdesc3 {
        num_rows    1
        num_columns  75
        variable     entdesc3, 0,0
        helpfile     help/entdesc.hlp
    }

    VARIABLE entdesc4 {
        foundia    "VARCHAR60 entdesc[3].arr"
        type       STRING
        format     4-60s
    }

    DATUM    entdesc4 {
        num_rows    1
        num_columns  75
        variable     entdesc4, 0,0
        helpfile     help/entdesc.hlp
    }

    TEXTLINE entdesc1txt ("Please enter a brief description for this assessment")

    DATUM    newassnam {
        num_rows    1
        num_columns  77
        variable     newassnam, 0,35
        leader       "Name of new assessment definition:"
        pickable     NO
    }

    WINDOW    newassdesc {
        num_rows    9
        num_columns  70
        datum       newassnam, 1,3
        textline     entdesc1txt, 3,4
        datum       entdesc1, 4,4
        datum       entdesc2, 5,4
        datum       entdesc3, 6,4
        datum       entdesc4, 7,4
        border       YES
    }

    -----
    -           Declarations for database howskeeping screen
    -----

    TEXTLINE updatetainfotxt ("Update information")

    BOTTOM    updatetainfo {
        num_rows    1
        num_columns  50
        textline     updatetainfotxt, 0,2
        helpfile     help/updatetain.hlp
    }

    TEXTLINE asstabletxt2 ("Print list of all columns in an assessment's tables")

    BOTTOM    asstable2 {
        num_rows    1
        num_columns  55
        textline     asstabletxt2, 0,2
        helpfile     help/asstable.hlp
    }

    TEXTLINE asstabletxt1 ("Print list of an assessment's tables")

    BOTTOM    asstable1 {
        num_rows    1
        num_columns  50
        textline     asstabletxt1, 0,2
        helpfile     help/asstable.hlp
    }

    TEXTLINE assess1txt ("Print list of all assessments")

    BOTTOM    assess {
        num_rows    1

```

```

num_columns 50
textline assessxt, 0,2
helpfile help/nohelp.hlp
}

TEXTLINE dbdattxt {"Print all database dates"}

BUTTON dbdata {
num_rows 1
num_columns 50
textline dbdattxt, 0,2
helpfile help/dbdata.hlp
}

TEXTLINE dbhsekgptxt {"WARNING: Actions you take on this screen affect
                        ASAM'S permanent databases!"}

WINDOW dbhsekgaction {
num_rows 13
num_columns 78
line 0,0, 0,77
textline dbhsekgptxt, 2,2
button dbdata, 4,2, "CALL dummy"
button assess, 6,2, "CALL prntabs"
button astable1, 8,2, "CALL $Uprpt 1"
button astable2, 10,2, "CALL $Uprpt 2"
button updateinfo, 12,2, "CALL dummy"
}

SCREEN dbhsekgpscreen {
title "DATABASE HOUSEKEEPING"
window dbhsekgaction, 2,1
window majoraction5, 18,1
border YES
}

WINDOW quickasbut {
num_rows 10
num_columns 2
button mabut00, 0, 0, "CALL $Uprnt deplmult[0].arr"
button mabut01, 1, 0, "CALL $Uprnt deplmult[1].arr"
button mabut02, 2, 0, "CALL $Uprnt deplmult[2].arr"
button mabut03, 3, 0, "CALL $Uprnt deplmult[3].arr"
button mabut04, 4, 0, "CALL $Uprnt deplmult[4].arr"
button mabut05, 5, 0, "CALL $Uprnt deplmult[5].arr"
button mabut06, 6, 0, "CALL $Uprnt deplmult[6].arr"
button mabut07, 7, 0, "CALL $Uprnt deplmult[7].arr"
button mabut08, 8, 0, "CALL $Uprnt deplmult[8].arr"
button mabut09, 9, 0, "CALL $Uprnt deplmult[9].arr"
}

WINDOW quickasbut1 {
num_rows 10
num_columns 2
button mabut10, 0, 0, "CALL $Uprnt deplmult[10].arr"
button mabut11, 1, 0, "CALL $Uprnt deplmult[11].arr"
button mabut12, 2, 0, "CALL $Uprnt deplmult[12].arr"
button mabut13, 3, 0, "CALL $Uprnt deplmult[13].arr"
button mabut14, 4, 0, "CALL $Uprnt deplmult[14].arr"
button mabut15, 5, 0, "CALL $Uprnt deplmult[15].arr"
button mabut16, 6, 0, "CALL $Uprnt deplmult[16].arr"
button mabut17, 7, 0, "CALL $Uprnt deplmult[17].arr"
button mabut18, 8, 0, "CALL $Uprnt deplmult[18].arr"
button mabut19, 9, 0, "CALL $Uprnt deplmult[19].arr"
}

TEXTLINE susaleact {"Print list of SUPERUSER's tables"}
TEXTLINE hqaleact {"Print list of HEADQUARTERS' tables"}

BUTTON susaleact {
num_rows 1
num_columns 50
textline susaleact, 0,2
helpfile help/nohelp.hlp
}

BUTTON hqaleact {
num_rows 1
num_columns 50
textline hqaleact, 0,2
helpfile help/nohelp.hlp
}

```

```

WINDOW retwopra {
    num_rows      1
    num_columns    76
    button        retassent, 0,1, "CALL ulistc", "NEW_SCREEN dbhsakpgscreen"
}

```

```

WINDOW hqrusaleact {
    num_rows      3
    num_columns    76
    button        hqrusaleact, 0,1, "CALL supropt 3"
    button        susaleact, 2,1, "CALL supropt 4"
}

```

```

SCREEN slclassscreen {
    title         "SELECT ASSESSMENT FOR DATABASE PRINTOUT"
    window        hqrusaleact, 4,1
    window        recalass, 8,1
    window        quickassbut, 9,3
    window        quickassbut1, 9,42
    window        cardat, 9,5
    window        cardat1, 9,44
    window        retwopra, 20,2
    border        YES
}

```

```

~      Declarations for view CHECK-LIST SCREEN

```

```

TEXTLINE otherchklist ("View another checklist")

```

```

BUTTON otherchklist {
    num_rows      1
    num_columns    25
    textline      otherchklist, 0, 2
    helpfile      help/othrchk.hlp
}

```

```

TEXTBLOCK fonsi2 {
    filename      txtblk/fonsi2.txt
    num_rows      10
    num_columns    76
    border        YES
}

```

```

WINDOW fonsi2 {
    num_rows      13
    num_columns    78
    textblock     fonsi2, 0,1
    button        otherchklist, 12,2, "REMOVE_WINDOW"
}

```

```

TEXTBLOCK fonsi1 {
    filename      txtblk/fonsi1.txt
    num_rows      10
    num_columns    76
    border        YES
}

```

```

WINDOW fonsi1 {
    num_rows      13
    num_columns    78
    textblock     fonsi1, 0,1
    button        otherchklist, 12,2, "REMOVE_WINDOW"
}

```

```

TEXTBLOCK ostar3 {
    filename      txtblk/ostar3.txt
    num_rows      10
    num_columns    76
    border        YES
}

```

```

WINDOW ontax3 {
    num_rows      13
    num_columns    78
    textblock      ontax3, 0,1
    button         otherchk1st, 12,2, "REMOVE_WINDOW"
}

TEXTBLOCK ontax2 {
    filename       txtblk/ontax2.txt
    num_rows       10
    num_columns     76
    border         YES
}

WINDOW ontax2 {
    num_rows      13
    num_columns    78
    textblock      ontax2, 0,1
    button         otherchk1st, 12,2, "REMOVE_WINDOW"
}

TEXTBLOCK ontax1 {
    filename       txtblk/ontax1.txt
    num_rows       10
    num_columns     76
    border         YES
}

WINDOW ontax1 {
    num_rows      13
    num_columns    78
    textblock      ontax1, 0,1
    button         otherchk1st, 12,2, "REMOVE_WINDOW"
}

TEXTLINE fonsi2txt {
    "Show documentation necessary for finding of no significant impact"}

BUTTON fonsi2 {
    num_rows      1
    num_columns    72
    textline       fonsi2txt, 0, 2
    helpfile       help/fonsi2.hlp
}

TEXTLINE fonsi1txt {
    "Show NEPA bases for finding of no significant impact (FONSI)"}

BUTTON fonsi1 {
    num_rows      1
    num_columns    65
    textline       fonsi1txt, 0, 2
    helpfile       help/fonsi1.hlp
}

TEXTLINE ontax3txt {"Show documentation needed for categorical exclusions"}

BUTTON ontax3 {
    num_rows      1
    num_columns    55
    textline       ontax3txt, 0, 2
    helpfile       help/ontax3.hlp
}

TEXTLINE ontax2txt {
    "Show examples of proposed actions qualifying for categorical exclusions"}

BUTTON ontax2 {
    num_rows      1
    num_columns    74
    textline       ontax2txt, 0, 2
    helpfile       help/ontax2.hlp
}

TEXTLINE ontax1txt {"Show NEPA bases for categorical exclusions (CAFEI)"}

BUTTON ontax1 {
    num_rows      1
    num_columns    55
    textline       ontax1txt, 0, 2
    helpfile       help/ontax1.hlp
}

```

```

WINDOW vwhklist {
    num_rows 11
    num_columns 78
    line 0,0 , 0,77
    button cstat1, 2,2, "ADD_WINDOW cstat1 3 1"
    button cstat2, 4,2, "ADD_WINDOW cstat2 3 1"
    button cstat3, 6,2, "ADD_WINDOW cstat3 3 1"
    button fons11, 8,2, "ADD_WINDOW fons11 3 1"
    button fons12, 10,2, "ADD_WINDOW fons12 3 1"
}

```

```

SCREEN viewchecklistscreens {
    title "VIEW CHECKLIST"
    window vwhklist, 2,1
    window majoraction4, 18,1
    border YES
}

```

```

-----
SCREEN probetatscreens {
    title "ENVIRONMENTAL ASSESSMENT STATUS"
    mainscreen YES
    border YES
    window assessroom, 3,1
    window probetst, 7,1
    window chgassess, 16,1
    window majoraction, 18,1
}

```

A.3 Screen Description File for Report-Related Portions of ASAN

```

-----
~      Declarations for REPORT GENERATION SCREEN
-----

TEXTLINE viewnoisetxt ("View text on calculated noise effects for current assessment")

BUTTON viewnoisetx {
    num_rows 1
    num_columns 72
    textline viewnoisetxt, 0,2
    helpfile help/viewtext.hlp
}

TEXTLINE viewbplatetxt ("View boilerplate")

BUTTON viewbplate {
    num_rows 1
    num_columns 72
    textline viewbplatetxt, 0,2
    helpfile help/rolodex.hlp
}

TEXTLINE prnoisembptxt ("Print above text with associated boilerplate")

BUTTON prnoisembp {
    num_rows 1
    num_columns 72
    textline prnoisembptxt, 0,2
    helpfile help/rolodex.hlp
}

TEXTLINE prnoisetxt ("Print above text")

BUTTON prnoisetx {
    num_rows 1
    num_columns 72
    textline prnoisetxt, 0,2
    helpfile help/creatxt.hlp
}

```

```

TEXTLINE stdreptxt ("Print standard report")

BUTTON stdrept {
    num_rows 1
    num_columns 72
    textline stdreptxt, 0, 2
    helpfile help/creatst.hlp
}

TEXTLINE selectreptxt ("Actions you can now take to produce text and graphics:")

WINDOW reportaction {
    num_rows 15
    num_columns 77
    line 0, 1, 0, 76
    textline selectreptxt, 4, 1
    button stdrept, 6, 1, "CALL callrpt"
    button viewnoisetx, 8, 1, "CALL dummy"
    button prnoisetx, 10, 1, "CALL callrpt3"
    button prnoisewp, 12, 1, "CALL callrpt4"
    button viewbplate, 14, 1, "NEW_SCREEN bplateareviewscreen"
}

SCREEN reportgenscreen {
    title "MAKE A REPORT"
    mainscreen YES
    window reportaction, 2, 1
    border YES
}

-----
-      Declarations for boilerplate review screen
-----

TEXTLINE otherbplatetxt ("View other boilerplate text")

BUTTON otherbplate {
    num_rows 1
    num_columns 30
    textline otherbplatetxt, 0, 2
    helpfile help/birplate.hlp
}

TEXTBLOCK heardam {
    filename blrplt/bringdam.bpl
    num_rows 10
    num_columns 76
    border YES
}

WINDOW heardam {
    num_rows 13
    num_columns 78
    textblock heardam, 0, 1
    button otherbplate, 12, 2, "REMOVE_WINDOW"
}

TEXTLINE heardamtxt ("Hearing damage risk")

BUTTON heardam {
    num_rows 1
    num_columns 35
    textline heardamtxt, 0, 2
    helpfile help/birplate.hlp
}

TEXTBLOCK structdam {
    filename blrplt/strctdam.bpl
    num_rows 10
    num_columns 76
    border YES
}

WINDOW structdam {
    num_rows 13
    num_columns 78
    textblock structdam, 0, 1
    button otherbplate, 12, 2, "REMOVE_WINDOW"
}

TEXTLINE structdamtxt ("Structural damage")

```

```

BUTTON structdm {
    num_rows      1
    num_columns    35
    textline      structdmtxt, 0, 2
    helpfile      help/blrplate.hlp
}

TEXTBLOCK actintfer {
    filename      blrplt/elpintf.bpl
    num_rows      10
    num_columns    76
    border        YES
}

WINDOW actintfer {
    num_rows      13
    num_columns    78
    textblock     actintfer, 0,1
    button        otherbplate, 12,2, "REMOVE_WINDOW"
}

TEXTLINE actintfertxt ("Activity interference")

BUTTON actintfer {
    num_rows      1
    num_columns    35
    textline      actintfertxt, 0, 2
    helpfile      help/blrplate.hlp
}

TEXTBLOCK wildlife {
    filename      blrplt/wildlife.bpl
    num_rows      10
    num_columns    76
    border        YES
}

WINDOW wildlife {
    num_rows      13
    num_columns    78
    textblock     wildlife, 0,1
    button        otherbplate, 12,2, "REMOVE_WINDOW"
}

TEXTLINE wildlifetxt ("Wildlife")

BUTTON wildlife {
    num_rows      1
    num_columns    35
    textline      wildlifetxt, 0, 2
    helpfile      help/blrplate.hlp
}

TEXTBLOCK ocmintfer {
    filename      blrplt/spchintf.bpl
    num_rows      10
    num_columns    76
    border        YES
}

WINDOW ocmintfer {
    num_rows      13
    num_columns    78
    textblock     ocmintfer, 0,1
    button        otherbplate, 12,2, "REMOVE_WINDOW"
}

TEXTLINE ocmintfertxt ("Communication interference")

BUTTON ocmintfer {
    num_rows      1
    num_columns    35
    textline      ocmintfertxt, 0, 2
    helpfile      help/blrplate.hlp
}

TEXTBLOCK livestock {
    filename      blrplt/livestck.bpl
    num_rows      10
    num_columns    76
    border        YES
}

```



```

WINDOW livestock {
    num_rows      13
    num_columns    78
    textblock      livestock, 0,1
    button         otherplate, 12,2, "REMOVE_WINDOW"
}

TEXTLINE livestocktxt {"Economic damage to livestock"}

BUTTON livestock {
    num_rows      1
    num_columns    35
    textline       livestocktxt, 0, 2
    helpfile       help/blrplate.hlp
}

TEXTBLOCK anayace {
    filename       blrplt/anayace.hpl
    num_rows       10
    num_columns     76
    border         YES
}

WINDOW anacy {
    num_rows      13
    num_columns    78
    textblock      anayace, 0,1
    button         otherplate, 12,2, "REMOVE_WINDOW"
}

TEXTLINE anacytxt {"Human anacyace"}

BUTTON anacy {
    num_rows      1
    num_columns    35
    textline       anacytxt, 0, 2
    helpfile       help/blrplate.hlp
}

TEXTLINE bplatetxt {"Select one of the following to view available
noise effects boilerplate:"}

TEXTLINE returapt {"Return to report selection screen"}

BUTTON returapt {
    num_rows      1
    num_columns    42
    textline       returapt, 0, 2
    helpfile       help/nohelp.txt
}

WINDOW nulltext {
    num_rows      3
    num_columns    77
    button         returapt, 1, 1, "NEW_SCREEN reportgenscreen"
}

WINDOW bplate {
    num_rows      11
    num_columns    78
    line          0,0, 0,77
    textline       bplatetxt, 2,2
    button         anacy, 4,2, "ADD_WINDOW anacy 3 1"
    button         comintfer, 6,2, "ADD_WINDOW comintfer 3 1"
    button         actintfer, 8,2, "ADD_WINDOW actintfer 3 1"
    button         heardam, 10,2, "ADD_WINDOW heardam 3 1"
    button         livestock, 4,39, "ADD_WINDOW livestock 3 1"
    button         wildlife, 6,39, "ADD_WINDOW wildlife 3 1"
    button         structdam, 8,39, "ADD_WINDOW structdam 3 1"
}

SCREEN bplaterevscreen {
    title          "VIEW BOILERPLATE TEXT"
    window         bplate, 2,1
    window         nulltext, 10, 1
    border         YES
}

```

A.4 Screen Description File for Graphic Portion of ASAN

```
-----  
-INCLUDE STATEMENT FOR THE ASAN & GRAPHICS typedef DEFINITIONS NEEDED TO  
- COMPILE USERS.C WITHOUT INCORPORATING THE WHOLE OF THE COMPILER  
-----
```

```
INCLUDE ASANTYPE.H  
INCLUDE ASAN.H
```

```
-----  
- DECLARATIONS IN NON-LEXICAL OBJECT SEQUENCE .....  
-----
```

```
-----  
- Data structures for holding list of map names  
-----
```

```
VARIABLE dinam[0] {  
    type      STRING  
    format    4-8s  
}  
DATUM dinam0 {  
    num_rows   1  
    num_columns 8  
    variable   dinam[0] , 0 , 0  
    pickable   NO  
}  
  
VARIABLE dinam[1] {  
    type      STRING  
    format    4-8s  
}  
DATUM dinam1 {  
    num_rows   1  
    num_columns 8  
    variable   dinam[1] , 0 , 0  
    pickable   NO  
}  
  
VARIABLE dinam[2] {  
    type      STRING  
    format    4-8s  
}  
DATUM dinam2 {  
    num_rows   1  
    num_columns 8  
    variable   dinam[2] , 0 , 0  
    pickable   NO  
}  
  
VARIABLE dinam[3] {  
    type      STRING  
    format    4-8s  
}  
DATUM dinam3 {  
    num_rows   1  
    num_columns 8  
    variable   dinam[3] , 0 , 0  
    pickable   NO  
}  
  
VARIABLE dinam[4] {  
    type      STRING  
    format    4-8s  
}  
DATUM dinam4 {  
    num_rows   1  
    num_columns 8  
    variable   dinam[4] , 0 , 0  
    pickable   NO  
}  
  
VARIABLE dinam[5] {  
    type      STRING  
    format    4-8s  
}  
DATUM dinam5 {
```

```

    num_rows    1
    num_columns  8
    variable     diamn[5] , 0 , 0
    pickable     NO
}

VARIABLE diamn[6] {
    type        STRING
    format      4-8s
}
DATUM diamn6 {
    num_rows    1
    num_columns  8
    variable     diamn[6] , 0 , 0
    pickable     NO
}

VARIABLE diamn[7] {
    type        STRING
    format      4-8s
}
DATUM diamn7 {
    num_rows    1
    num_columns  8
    variable     diamn[7] , 0 , 0
    pickable     NO
}

VARIABLE diamn[8] {
    type        STRING
    format      4-8s
}
DATUM diamn8 {
    num_rows    1
    num_columns  8
    variable     diamn[8] , 0 , 0
    pickable     NO
}

VARIABLE diamn[9] {
    type        STRING
    format      4-8s
}
DATUM diamn9 {
    num_rows    1
    num_columns  8
    variable     diamn[9] , 0 , 0
    pickable     NO
}

VARIABLE diamn[10] {
    type        STRING
    format      4-8s
}
DATUM diamn10 {
    num_rows    1
    num_columns  8
    variable     diamn[10] , 0 , 0
    pickable     NO
}

VARIABLE diamn[11] {
    type        STRING
    format      4-8s
}
DATUM diamn11 {
    num_rows    1
    num_columns  8
    variable     diamn[11] , 0 , 0
    pickable     NO
}

VARIABLE diamn[12] {
    type        STRING
    format      4-8s
}
DATUM diamn12 {
    num_rows    1
    num_columns  8
    variable     diamn[12] , 0 , 0
    pickable     NO
}

```

```

VARIABLE d1am[13] {
    type      STRING
    format    4-8s
}
DATUM d1am13 {
    num_rows  1
    num_columns 8
    variable   d1am[13] , 0 , 0
    pickable   NO
}

VARIABLE d1am[14] {
    type      STRING
    format    4-8s
}
DATUM d1am14 {
    num_rows  1
    num_columns 8
    variable   d1am[14] , 0 , 0
    pickable   NO
}

-----
--          Declaration for entering coordinates
-----

VARIABLE entlat {
    foundin    "COORDINATE ent.lat"      ~ CURRENT or ENTER Set
    type      STRING
    format    4-13s
}
DATUM entlat {
    num_rows  1
    num_columns 25
    variable   entlat, 0 ,11
    leader     "Latitude: "
    helpfile   help/contmap.hlp
}

VARIABLE entlong {
    foundin    "COORDINATE ent.long"
    type      STRING
    format    4-13s
}
DATUM entlong {
    num_rows  1
    num_columns 25
    variable   entlong, 0 ,11
    leader     "Longitude:"
    helpfile   help/contmap.hlp
}

-----
--          Declaration for show coordinates
-----

VARIABLE shwlat {
    foundin    "COORDINATE show.lat"
    type      STRING
    format    4-13s
}
DATUM shwlat {
    num_rows  1
    num_columns 25
    variable   shwlat, 0 ,11
    leader     "Latitude: "
    pickable   NO
    helpfile   help/contmap.hlp
}

VARIABLE shwlong {
    foundin    "COORDINATE show.long"
    type      STRING
    format    4-13s
}
DATUM shwlong {
    num_rows  1
    num_columns 25

```

```

        variable shwlong, 0, 11
        leader "Longitude:"
        pickable NO
        helpfile help/combmap.hlp
    }

    VARIABLE shwdist {
        type DOUBLE
        uplimit 10000.
        lowlimit 0.
        format %10.2lf
    }

    DATUM shwdist {
        num_rows 1
        num_columns 13
        variable shwdist, 0, 0
        pickable NO
        trailer "km"
    }

-----
- DECLARATIONS FOR SCREEN HEADER
-----

    VARIABLE asseaname {
        foundin "ASAMHEADR ASSESSMENT.name"
        type STRING
        format %-30s
    }

    DATUM asseaname1 {
        num_rows 1
        num_columns 78
        variable asseaname, 0, 30
        leader "Name of current assessment: "
        pickable NO
        helpfile help/nohelp.hlp
    }

    VARIABLE comment {
        foundin "ASAMHEADR ASSESSMENT.desc"
        type STRING
        format %-66s
    }

    DATUM comment {
        num_rows 1
        num_columns 77
        variable comment, 0, 10
        pickable NO
    }

    WINDOW lia {
        num_rows 1
        num_columns 78
        line 0, 0, 0, 77
    }

    WINDOW asseanamecom {
        num_rows 3
        num_columns 78
        datum asseaname1, 0, 0
        datum comment, 1, 1
        line 2, 0, 2, 77
    }

-----
- Declaration for buttons used in majoraction footer
-----

    TEXTLINE probeta { "REVIEW CURRENT ASSESSMENT STATUS" }

    BUTTON probetat {
        num_rows 1
        num_columns 35
        textline probeta, 0, 2
        helpfile help/nohelp.hlp
    }

    TEXTLINE probetal { "WORK ON ANOTHER ASSESSMENT" }

```

```

BUTTON probetat1 {
    num_rows      1
    num_columns    28
    textline       probetat1, 0, 2
    helpfile       help/mchelp.hlp
}

TEXTLINE probeta2 { "REVIEW ASSESSMENT STATUS" }

BUTTON probetat2 {
    num_rows      1
    num_columns    26
    textline       probeta2, 0, 2
    helpfile       help/mchelp.hlp
}

TEXTLINE probdef { "ADD INFORMATION TO CURRENT ASSESSMENT" }

BUTTON probdef {
    num_rows      1
    num_columns    39
    textline       probdef, 0, 2
    helpfile       help/probdef.hlp
}

TEXTLINE probdef1 { "ADD TO ASSESSMENT DEFINITION" }

BUTTON probdef1 {
    num_rows      1
    num_columns    31
    textline       probdef1, 0, 2
    helpfile       help/probdef.hlp
}

TEXTLINE analysis { "ANALYZE DATA" }

BUTTON analysis {
    num_rows      1
    num_columns    15
    textline       analysis, 0, 2
    helpfile       help/datanal.hlp
}

TEXTLINE reportgen { "MAKE A REPORT" }

BUTTON reportgen {
    num_rows      1
    num_columns    16
    textline       reportgen, 0, 2
    helpfile       help/reportgen.hlp
}

TEXTLINE vchklis { "VIEW CHECKLIST FOR CURRENT ASSESSMENT" }

BUTTON vchklis {
    num_rows      1
    num_columns    42
    textline       vchklis, 0, 2
    helpfile       help/viewlist.hlp
}

TEXTLINE databasinq { "MAKE DATABASE INQUIRIES" }

BUTTON databasinq {
    num_rows      1
    num_columns    26
    textline       databasinq, 0, 1
    helpfile       help/making.hlp
}

-----
-      Declarations for majoraction footers
-----

WINDOW majoraction {      -header for problem status screen
    num_rows      4
    num_columns    78
    line          0, 0, 0, 77
    title         "Please select one of the following actions:"
    button        probdef,      2, 0, "CALL dummy"
    button        analysis,     2, 43, "CALL dummy"
    button        reportgen,    2, 62, "CALL dummy"
}

```

```

        button      vwebklist, 3, 0, "CALL dummy"
        button      databaseinq, 3, 52, "CALL dummy"
    }

WINDOW majoraction3 {      -header for report generation screen
    num_rows      4
    num_columns    78
    line          0, 0, 0, 77
    title         "Alternative actions you can now take:"
    button        probstat2, 2, 0, "CALL dummy"
    button        probdef1, 2, 29, "CALL dummy"
    button        vwebklist, 3, 0, "CALL dummy"
    button        databaseinq, 3, 52, "CALL dummy"
}

```

```

-      Declarations for MAP CONTROL

```

```

VARIABLE shwareaLDW {
    type          INTEGER
    format        4-5d
    uplimit       124
    default       75
    lowlimit      20
}

```

```

DATUM shwareaLDW {
    num_rows      1
    num_columns    35
    variable      shwareaLDW, 0, 15
    leader        "Show areas of "
    trailer       "(dB) <= LDW <="
    helpfile      help/nchelp.hlp
}

```

```

VARIABLE shwareaLDW1 {
    type          INTEGER
    format        4-5d
    uplimit       124
    default       75
    lowlimit      20
}

```

```

DATUM shwareaLDW1 {
    num_rows      1
    num_columns    18
    variable      shwareaLDW1, 0, 0
    trailer       "(dB)"
    helpfile      help/nchelp.hlp
}

```

```

VARIABLE shwareaPSF {
    type          INTEGER
    format        4-5d
    uplimit       124
    default       75
    lowlimit      20
}

```

```

DATUM shwareaPSF {
    num_rows      1
    num_columns    35
    variable      shwareaPSF, 0, 15
    leader        "Show areas of "
    trailer       "(dB) <= PSF <="
    helpfile      help/nchelp.hlp
}

```

```

VARIABLE shwareaPSF1 {
    type          INTEGER
    format        4-5d
    uplimit       124
    default       75
    lowlimit      20
}

```

```

DATUM shwareaPSF1 {
    num_rows      1
    num_columns    18
    variable      shwareaPSF1, 0, 0
    trailer       "(dB)"
}

```

```

        helpfile    help/achelp.hlp
    }

TEXTLINE    entidptofint    ("Mark the map at:")

TEXTLINE    shwidptofint    ("Locate touched point")

BUTTON      shwidptofint {
    num_rows    1
    num_columns    22
    textline    shwidptofint, 0, 2
    helpfile    help/achelp.hlp
}

TEXTLINE    shwdist    ("Show distance between two touched points:")

BUTTON      shwdist {
    num_rows    1
    num_columns    43
    textline    shwdist, 0, 2
    helpfile    help/achelp.hlp
}

TEXTLINE    addalmap    ("Add or Remove layers being displayed")

BUTTON      addalmap {
    num_rows    1
    num_columns    77
    textline    addalmap, 0, 2
    helpfile    help/mapname.hlp
}

TEXTLINE    addalmapinfo    ("Add or Delete information on a map layer")

BUTTON      addalmapinfo {
    num_rows    1
    num_columns    77
    textline    addalmapinfo, 0, 2
    helpfile    help/addalmap.hlp
}

TEXTLINE    showlegend    ("Show the legend" )

BUTTON      showlegend {
    num_rows    1
    num_columns    17
    textline    showlegend, 0, 2
    helpfile    help/achelp.hlp
}

TEXTLINE    hidalegend    ("Hide the legend" )

BUTTON      hidalegend {
    num_rows    1
    num_columns    17
    textline    hidalegend, 0, 2
    helpfile    help/achelp.hlp
}

TEXTLINE    erasedisplay    ("Erase the map display" )

BUTTON      erasedisplay {
    num_rows    1
    num_columns    77
    textline    erasedisplay, 0, 2
    helpfile    help/achelp.hlp
}

TEXTLINE    editcolors    ("Edit the color assignments" )

BUTTON      editcolors {
    num_rows    1
    num_columns    77
    textline    editcolors, 0, 2
    helpfile    help/achelp.hlp
}

VARIABLE    layersavename {
    type    STRING
    format    %s
}

```



```

DATUM layersaveame {
    num_rows 1
    num_columns 77
    variable layersaveame, 0, 35
    leader "Save the displayed map as:"
    helpfile help/saveamp.hlp
}

VARIABLE vpsame {
    type STRING
    format %7s
}

DATUM vpsame {
    num_rows 1
    num_columns 17
    leader "view ("
    trailer ");:"
    variable vpsame, 0, 7
    pickable NO
}

TEXTLINE vw_c { "Sells.C" }

BUTTON vw_c {
    num_rows 1
    num_columns 9
    textline vw_c, 0, 2
}

TEXTLINE vw_ma { "Ajo.M" }

BUTTON vw_ma {
    num_rows 1
    num_columns 7
    textline vw_ma, 0, 2
}

TEXTLINE vw_ms { "Sells.M" }

BUTTON vw_ms {
    num_rows 1
    num_columns 9
    textline vw_ms, 0, 2
}

TEXTLINE vw_fa { "Ajo.F" }

BUTTON vw_fa {
    num_rows 1
    num_columns 7
    textline vw_fa, 0, 2
}

TEXTLINE vw_fs { "Sells.F" }

BUTTON vw_fs {
    num_rows 1
    num_columns 9
    textline vw_fs, 0, 2
}

WINDOW mapotrlaction {
    num_rows 13
    num_columns 78
    datum vpsame, 0, 1
    button vw_c, 0, 21, "CALL new_view 'Sells.C'"
    button vw_ma, 0, 33, "CALL new_view 'Ajo.M'"
    button vw_ms, 0, 43, "CALL new_view 'Sells.M'"
    button vw_fa, 0, 55, "CALL new_view 'Ajo.F'"
    button vw_fs, 0, 65, "CALL new_view 'Sells.F'"
    button addalmap, 1, 1, "NEW_SCREEN mapman"
    button addalmapinfo, 2, 1, "NEW_SCREEN addalmapinfo"
    button showlegend, 3, 1, "CALL show_legend"
    button hidellegend, 3, 30, "CALL hide_legend"
    button erasedisplay, 4, 1, "CALL clear_screen"
    textline editdptofint, 5, 3
    datum editlat, 5, 21, "CALL lat2dec fcent", "NEWVALS"
    datum editlong, 5, 48, "CALL lon2dec fcent", "NEWVALS"
    button editcolors, 6, 1, "CALL edit_colors"
    button shwidptofint, 7, 1, "CALL show_coords"
    datum shwlat, 7, 25

```

```

        datum    shulocg,      7, 52
        button   shwdist,      8, 1, "CALL show_dist"
        datum    shwdist,      8, 46
        datum    shwareaLDW,   9, 3, "CALL dummy"
        datum    shwareaLDW1,  9, 40, "CALL dummy"
        datum    shwareaPSF,   10, 3, "CALL dummy"
        datum    shwareaPSF1,  10, 40, "CALL dummy"
        datum    layersaveama, 12, 1, "CALL store_screen"
    }

SCREEN mapcontrol {
    title        "MAP DISPLAY CONTROL"
    mainscreen   YES
    window       asnamcon, 2,1
    window       mapctrlaction, 5,1
    window       mapaction3, 16,1
    border       YES
}

-----
~      Declaration for MAP MANAGER
-----

DATUM curview {
    num_rows     1
    num_columns   22
    leader       "CURRENT VIEW:"
    variable      vpsama, 0, 14
    pickable     NO
}

TEXTLINE layertxt_tit { "AVAILABLE MAP LAYERS:" }

TEXTBLOCK layertxt {
    num_rows     12
    num_columns   51
    border       YES
    filename     layers.txt
}

TEXTLINE layernow_tit { "NOW DISPLAYED:" }

WINDOW layertxt {
    num_rows     14
    num_columns   51
    datum        curview, 0, 0
    textline     layertxt_tit, 1, 0
    textline     layernow_tit, 0, 37
    textblock    layertxt, 2, 0
}

WINDOW layernow {
    num_rows     17
    num_columns   10
    border       YES
    datum        dlnam0, 1, 1
    datum        dlnam1, 2, 1
    datum        dlnam2, 3, 1
    datum        dlnam3, 4, 1
    datum        dlnam4, 5, 1
    datum        dlnam5, 6, 1
    datum        dlnam6, 7, 1
    datum        dlnam7, 8, 1
    datum        dlnam8, 9, 1
    datum        dlnam9, 10, 1
    datum        dlnam10, 11, 1
    datum        dlnam11, 12, 1
    datum        dlnam12, 13, 1
    datum        dlnam13, 14, 1
    datum        dlnam14, 15, 1
}

VARIABLE layer2add {
    type         STRING
    format       %s
}

DATUM layer2add {
    num_rows     1
    num_columns   34
    leader       "Add map layer"
    trailer      "to display"
}

```

```

        variable    layer2add , 0 , 15
    }

VARIABLE layer2del {
    type    STRING
    format    48s
}

DIALOG layer2del {
    num_rows    1
    num_columns    39
    leader    "Remove Map Layer"
    trailer    "from display"
    variable    layer2del , 0 , 18
}

WINDOW asklayer {
    num_rows    2
    num_columns    41
    datum    layer2add , 0 , 1 ,    "CALL add_layer"
    datum    layer2del , 1 , 1 ,    "CALL del_layer"
}

TEXTLINE done1 {"Return to Map Control Screen"}

BUTTON done1 {
    num_rows    1
    num_columns    33
    textline    done1, 0 , 2
    helpfile    help/newsamst.hlp
}

WINDOW done1 {
    num_rows    1
    num_columns    50
    button    done1, 0 , 2,    "NEW_SCREEN mapcontrol"
    helpfile    help/newsamst.hlp
}

SCREEN mapman {
    title    "MAP SCREEN MANAGEMENT"
    window    asnamocm , 2 , 1
    window    layartxt , 5 , 2
    window    layarnow , 5 , 54
    window    asklayer , 19 , 1
    window    done1 , 21 , 1
    border    YES
}

-----
-   Declarations for ADD OR DELETE INFORMATION FROM A MAP
-----

TEXTLINE notavail {"Sorry, this facility is not yet available"}

WINDOW adddelmapinfo {
    num_rows    5
    num_columns    78
    textline    notavail, 4,10
    helpfile    help/nohelp.hlp
}

SCREEN adddelmapinfo {
    title    "ADD OR DELETE INFORMATION ON A MAP LAYER"
    window    asnamocm, 2, 1
    window    adddelmapinfo, 5, 1
    window    done1 , 14 , 15
    window    majoraction, 18,1
    border    YES
}

-----
-   Declaration for entering coordinates window
-----

TEXTLINE entupplf
    {"Enter upper-left corner coordinates of area of current interest"}

TEXTLINE entlowrt
    {"Enter lower right corner coordinates of area of current interest"}

```

```

DATUM shvlat2 {
    num_rows 1
    num_columns 25
    variable shvlat, 0,11
    leader "Latitude: "
    helpfile help/combmap.hlp
}

DATUM shvlong2 {
    num_rows 1
    num_columns 25
    variable shvlong, 0,12
    leader "Longitude: "
    helpfile help/combmap.hlp
}

WINDOW entocor {
    num_rows 10
    num_columns 78
    textline entupplf, 2, 2
    datum entlat, 3, 1, "CALL lat2dec &ent", "NEWVALS"
    datum entlong, 3, 36, "CALL lon2dec &ent", "NEWVALS"
    textline entlowrt, 8, 2
    datum shvlat2, 6, 1, "CALL lat2dec &show", "NEWVALS"
    datum shvlong2, 6, 36, "CALL lon2dec &show", "NEWVALS"
}

- end of g.edf

```

Appendix B

PROGRAM LISTINGS

The following listings are the C-language software modules of which ASAN is composed.

```

/*
 *      asan.pc -- ASAN Main Program
 *
 *      1. Opens the printer.
 *      2. Calls strtASAN to establish communication with ORACLE
 *      3. Verifies that ASAN software is valid.
 *      4. Calls Uinit to start the screen driver.
 *      5. At end of the session closes database and printer
 *
 */
*****

#include <stdio.h>
#include <process.h>
#include <string.h>
#include <time.h>

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE sbarris.h;
EXEC SQL INCLUDE hostvars.h;

EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE SQLCA;
#define SQLCA_STORAGE_CLASS
#include "asan.h"

main()
{
extern int BBA_DEBUG_FEATURES;

int i;
int OC_u2ll_etheroid(), closeORA(), logentry(), strtASAN(), vfy_ASAN();
void aspMsg();

static char *legal_notice[11] =
{"\n\n\n\t\t\t RESTRICTED RIGHTS LEGEND\n",
"\tUse, duplication, or disclosure is subject to restrictions\n",
"\t\t\t as set forth in subdivision (b) (3) (ii) of the\n",
"\t\t\t Rights in Technical Data and Computer Software Clause\n",
"\t\t\t at 52.227-7013 of the DOD FAR Supplement.\n",
"\t\t\tBBN LABORATORIES INCORPORATED\n",
"\t\t\t\t\t 10 MOULTON STREET\n",
"\t\t\t\t\t Cambridge, MA 02238\n",
"\t\t\t\t\t 617-873-3000\n",
"\t\t\t User Interface Copyright (C) 1985, BBN Laboratories Incorporated\n",
"\t\t\t All Rights Reserved"};

FILE *fopen();

pra = fopen("pra","a");
if (pra == NULL) {
printf("\nCan't open printer!"); exit(128); }
dante = fopen("chronfil.saf","a");
if (dante == NULL) {
printf("\nCan't open chronfile!"); exit(128); }

printf("%sJ",27);
for (i = 0; i < 11; i++) printf("%s",legal_notice[i]);

printf("\n\n\n\t\t\t Please tap the space bar to continue, \"CTRL-C\" \n\n\n\t\t\t to abort.\n");
do {
i = getch();
} while (i != ' ');

printf("\nasan starting.....");

```

```

ORA_DEBUG_FEATURES = 1;

utmsize = 12;
OC_u2ll_spheroid("clark66");

startASAN();
#ifdef CHECKOUT
printf("\nsuccessfully connected as Project %d (%s)\nPrivileges are",
      ASSESSMENT.id, ASSESSMENT.name);
for (i = 0; i < 3; i++) printf(" %s ", ASSESSMENT.auth[i]);
#endif

/* You are now successfully CONNECTed to ORACLE. Compare the */
/* software actually running to what is in the validation file. */

if (vfy_ASAN()) exit(16);

/* You are now successfully CONNECTed to ASAN. Control is */
/* transferred to Vinit, whence it returns only at the very */
/* end unless something untoward happens along the way and */
/* ASAN decides to pull the plug in midstream somewhere. */

Vinit();
logentry();
closeORA();
printf("%c{24;12",27);
exit(0);
}

/*****
 *
 * oca2db.pc -- Routines connecting a USERNAME to ORACLE, etc.
 *
 * This file contains:
 *
 * ASANocora - connects any ASAN Assessment to ORACLE
 * closeORA - closes ORACLE DMS
 * lastsess - retrieves last session from the logbook
 * logentry - makes an entry in the Assessment's logbook
 * myOrname - fills in ORACLE user portion of ASSESSMENT structure
 * enrollORA - enrolls a new Assessment as an ORACLE User
 * SUocora - connects ASAN's SUPERUSER to ORACLE.
 * vfyoid - determines if a username exists in ORACLE.
 *
 *****/

#include <process.h> /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */
EXEC SQL BEGIN DECLARE SECTION; /* All SQL declarations for this */
EXEC SQL INCLUDE hostvars.h; /* are in these two header files */
EXEC SQL INCLUDE subarrts.h; /* this one comes from "U" */
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asan.h" /* Standard ASAN Header File */

int ASANocora(name)
/*****
 *
 * ASANocora -- Connect an ASAN Assessment to ORACLE
 *
 * Routine ROLLS BACK any outstanding transactions of the current
 * user and then connects to the requested user. Returns an error
 * code which indicates whether or not the CONNECT was successful.
 *
 * Notes: 1. The existence of the assessment on the database is
 * assumed. ( vfyoid(name) is available to check.)
 *
 * 2. If the name requested is not a "real" assessment
 * but one of the privileged ORACLE names (used for
 * system maintenance, DBA functions, etc.) the CONNECT
 * will fail (sqlca.sqlcode == SQL_RAD_LOGIN). If so,
 * you will be re-CONNECTed to the old assessment and
 * the error code of the failed connect is returned
 * to the calling program.
 *
 *****/

```

```

*****/

char name[];
{
    register int i, j;
    int temp, logentry(), loggedon(), myCname(), recon, success();
    void expMsg();
    char *clock();

#ifdef CHECKOUT
    printf("\nASAWconn: %s", name);
#endif

loggedon();          /* Verify that we know who the person responsible is */
SLOUT("Please stand by: Switching assessments....");
logentry();          /* Make entry in log book and disconnect previous user */
WHEREAMI(screen, Window, Datum, Button);
misslabl.arr[0] = misdesc.arr[0] = '\0';
misslabl.len = misdesc.len = 0;
if ( (sqlca.sqlcode == 0) || /* These are reasonable return codes */
    (sqlca.sqlcode == NOT_LOGGED_ON) ||
    (sqlca.sqlcode == NOT_CONNECTED) ) {
    strcpy ( uid.arr, name ); /* Note: The SQL CONNECT statement */
    uid.len = strlen ( uid.arr ); /* doesn't like an immediate password */
    strcpy ( pwd.arr, univpwd ); /* and a dynamic username..... */
    pwd.len = strlen ( pwd.arr );
    EXEC SQL CONNECT uid IDENTIFIED BY :pwd;
    temp = (int) sqlca.sqlcode;
#ifdef CHECKOUT
    printf(" Connect = %ld", sqlca.sqlcode);
#endif

    switch(temp) {
        case 0: /* What did ORACLE come up with? */
            /* CONNECT succeeded as planned */
            /* Load the ASSESSMENT structure */
            /* and if that was o.k. you can */
            /* Pick up the description from */
            /* the site's table of contents */
            if (myCname()) {
                EXEC SQL SELECT description
                FROM table_of_contents
                INTO :workspace
                WHERE idnumber = :username;
                #ifdef CHECKOUT
                printf(" %ld", sqlca.sqlcode);
                #endif
                if (!sqlca.sqlcode) { /* Found a description in table of contents */
                    workspace.arr[workspace.len] = '\0'; /* Truncate description */
                    j = (sizeof ASSESSMENT.desc) - 1; /* to fit on the screen */
                    if (workspace.len >= j) {
                        for(i = 0; i < j; i++) ASSESSMENT.desc[i] = workspace.arr[i];
                        ASSESSMENT.desc[j] = '\0';
                    }
                    else strcpy(ASSESSMENT.desc, workspace.arr);
                }
                else { /* No Description..... How can this be???? */
                    expMsg;
                    printf(dante, "\n%s ASAWconn: %s not in T.O.C", clock(), uid.arr);
                    printf(dante, "\n\t\t\t %s", sqlca.sqlwarn.sqlwarn);
                    SLOUTS("Security Violation: Table of contents error");
                }
            }
            /* This will become a security violation! */

            closeORA();
            exit(255); /* But for now, let it go by */

            strcpy(ASSESSMENT.desc,
                "Description missing from table of contents");
        }

        EXEC SQL SELECT TO_CHAR(SYSDATE, 'DD-Mon-YY HH24:MI:SS')
        FROM dual INTO :timet1;
        #ifdef CHECKOUT
        printf("%ld", sqlca.sqlcode);
        #endif

        EXEC SQL SELECT TO_CHAR(start_work, 'dd-Mon-yy HH24:MI:SS')
        FROM logbook
        INTO :startdate
        WHERE start_work =
            (SELECT MIN(start_work) from logbook);
    }
}

```

```

#ifdef CHECKOUT
printf("%ld", sqlca.sqlcode);
#endif

if (sqlca.sqlcode) {
    /* If you are in the T of C */
    if (sqlca.sqlcode == SQL_MGR) { /* but not here, you are new */
        strcpy(startdate.arr, timest1.arr, timest1.len);
        startdate.len = timest1.len;
        strcpy(lastdate.arr, timest1.arr, timest1.len);
        lastdate.len = timest1.len;
        lastmtr.len = 0;
        lastmca.len = 0;
    }
    else {
        SENDMSG;
        sprintf(dante, "\n%s ASAWconn: %s logbook error",
            clock(), uid.arr);
        sprintf(dante, "\n\t\t\t %s", sqlca.sqlarm.sqlarmc);
    }
}
else {
    EXEC SQL SELECT lastmtr, lastmca
        FROM logbook
        INTO :lastmtr, :lastmca
        WHERE stop_work =
            (SELECT MAX(stop_work) from logbook);
#ifdef CHECKOUT
printf("%ld", sqlca.sqlcode);
#endif
}
lastmtr.arr[lastmtr.len] = '\0';
lastmca.arr[lastmca.len] = '\0';

/* All that's left now is to put up the screen */
NEW_SCREEN("probatatscreen");
NEWVALS();
if (ASSESSMENT.auth[2] != 'Y')
    SLOUTP("This assessment does not have resource authorization");
return (int) 0;
}
else {
    SENDMSG;
    sprintf(dante, "\n%s Mysterious failure on %s", clock(), uid.arr);
    sprintf(dante, "\n\t\t\t %s", sqlca.sqlarm.sqlarmc);
    sprintf(dante, "\n\t\t\t while retrieving info from SYS.VARNOUSER");
    temp = (int) sqlca.sqlcode;
    SLOUTP("Undiagnosible system failure. ASAW will restart");
    strcpy(ASSESSMENT.name, "SUPERUSER");
    NEW_SCREEN("firstscreen");
    break;
}

/* Trouble in River City if you get this far down in the switch .... */

case SQL_NO_USERNAME:
    SLOUTP("You cannot use a blank as a name");
    break;

case SQL_BAD_LOGIN:
case NOT_LOGGED_ON:
    SENDMSG;
    sprintf(dante, "\n%s Authorization failure on %s", clock(), uid.arr);
    sprintf(dante, "\n\t\t\t %s", sqlca.sqlarm.sqlarmc);
    sprintf(workspace.arr, "Unsuccessful: %s",
        sqlca.sqlarm.sqlarmc);
    SLOUTP(workspace.arr);
    break;

default:
    SENDMSG;
    sprintf(dante, "\n%s Unsuccessful login: %s", clock(), uid.arr);
    sprintf(dante, "\n\t\t\t ", sqlca.sqlarm.sqlarmc);
    sprintf(workspace.arr, "Unsuccessful: %s", sqlca.sqlarm.sqlarmc);
    SLOUTP(workspace.arr);
} /* End SWITCH */

strcpy (uid.arr, ASSESSMENT.name); /* Re-CONNECT with */
uid.len = strlen (uid.arr); /* previous name. */
if ( strcmp("SUPERUSER", uid.arr) ) {
    EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
    reccn = (int) sqlca.sqlcode;
    /* Re-initialize the screen. Logentry() call has closed all cursors! */
    if (!reccn || (strcmp(screen, "chgcurasscreen")) ) {
        penhase();
    }
}

```



```

        return temp; }
    }
    else recon = SUCCES();

    if (recon) { /* How we're stuck beyond help */
        MSGMSG;
        SLOUTMSG("ASAM got \"stuck\" looking for the data.....");
        sprintf(dante, "\n%s ASAMoon stuck: %s", clock(),
            sqlca.sqlwarn.sqlwarn);
        sprintf(dante, "\n\t\t\t\t\t After attempted re-connect to %s", uid.usr);
        closeORA();
        exit(255);
        SLOUTMSG("ASAM must restart due to an internal error");
        MSG_SCREEN("firstscreen");
        return temp;
    }

    else { expmsg(); /* If you get here, you've been doing */
        sprintf(dante, /* something outstandingly bizarre a */
            "\n%s Forced shut-down by %s", /* long time before this function was */
            clock(), /* called. No idea how to recover! */
            sqlca.sqlwarn.sqlwarn);
        closeORA();
        exit(255);
    }
}

closeORA()
/*****
 *
 *      closeORA -- Disconnect an ASAM Assessment from ORACLE
 *      ----- and return ORACLE to original state as
 *              given by OO_roode.
 *
 *      Routine ROLLS BACK any outstanding transactions of the current
 *      user and then disconnects from ORACLE. Returns an error code
 *      which indicates whether or not the call was successful.
 *
 *
 *      Notes:  1. The value of OO_code is set in the main() function
 *                as the return code of the initial logon attempt. IF
 *                these codes change in subsequent releases of ORACLE
 *                this function must be recompiled with the new codes.
 *
 *                2. Unless the current user is SUPERUSER or this function
 *                is called because of an unrecoverable ORACLE error,
 *                calls to closeORA() should be preceded by logentry();
 *
 *****/
{
    int doode, fclose(), spawnlp();
    char *clock();

#ifdef CHECKOUT
    printf("\n\ncloseORA");
#endif

    SLOUT("          *** End ASAM ***");
    sprintf(dante, "\n%s ASAM Close-down process:", clock());

    MSG MSG ROLLBACK WORK RELEASE;
#ifdef CHECKOUT
    printf(" rolled back tld", sqlca.sqlcode);
#endif

    if (OO_roode == 0) { /* We're done if ORACLE was up when we started */
        sprintf(dante, " Servers retained intact");
        return 0;
    }

    if ((OO_roode == ORA_UNAVAILABLE) || (OO_roode == -3120)) {
        doode = spawnlp(P_WAIT, "ior.exe", "ior.exe", "shut", NULL);
#ifdef CHECKOUT
        printf(" %d", doode);
#endif
        sprintf(dante, "\n%s Shut down ORACLE Server (%d)", clock(), doode);
        if (OO_roode != ORA_UNAVAILABLE) {
            doode = spawnlp(P_WAIT, "remora.exe", "remora.exe", "all", NULL);
#ifdef CHECKOUT
            printf(" %d", doode);
#endif
            sprintf(dante, "\n%s Deinstalled SQL*Plus (%d)", clock(), doode);
            return doode;
        }
    }
}

```

```

    }
    else { /* Could be ASAN wasn't installed, or ..... */
        sprintf(dante, " Servers left unchanged (Code %d)", CO_roots);
    }
    sprintf(dante, "\n");
    fclose(dante);
    fclose(prn);
}

int lastsess()
/*****
 *
 * lastsess -- Retrieves last entry from the ASAN logbook. It is
 *             used when we need to establish the last activity on
 *             the system. (SUPERUSER does not log its activity.)
 *
 * Returns the value of sqlca.sqlcode to calling program
 *
 *****/
{
    char *clock();

#ifdef CHECKOUT
    printf("\nlastsess ");
#endif

    SROUT("Re-loading the last session's parameters");
    EXEC SQL SELECT planner, TO_CHAR(stop_work, 'DD-Mon-YY HH24:MI:SS'), idnumber
    /*      lastlog */
    FROM    last_login
    INTO    :planlist, :lastdate, :username;

#ifdef CHECKOUT
    printf("%ld", sqlca.sqlcode);
#endif

    if (!sqlca.sqlcode) { /* This is what happens when all is O.K. */
        planlist.arr[planlist.len] = '\0';
        lastdate.arr[lastdate.len] = '\0';
    }

    else if ( sqlca.sqlcode == SQL_EOF ) { /* When you start the very first time
    */
        strcpy(planlist.arr, "No work ever done yet");
        planlist.arr[21] = '\0';
        planlist.len = 21;
        lastdate.arr[0] = '\0';
        lastdate.len = 0;
    }

    else {
        SROMSG;
        sprintf(dante,
            "\n%s Incomprehensible return code %ld retrieving last session",
            clock(), sqlca.sqlcode);
        sprintf(dante, "\n\t\t\t %s", sqlca.sqlwarn.sqlwarnm);
        expMsg();
    }

#ifdef CHECKOUT
    printf("\n%s, %s, %d", planlist.arr, lastdate.arr, username);
#endif
    return (int) sqlca.sqlcode;
}

int logentry()
/*****
 *
 * logentry -- Makes an entry in the ASAN register for the work just
 *             performed on the assessment. It is called by the
 *             ASANConn and SROConn routines when they close out the
 *             previous assessment.
 *
 * Returns the value of sqlca.sqlcode to calling program
 *
 *****/
{

```

```

int rooda;
char *clock();

#ifdef CHECKOUT
    printf("\alogentry: ");
#endif

if (strcmp(ASSESSMENT.name, "SUPERUSER")) { /* Make entry in log book if the */
                                           /* previous name was not SUPERUSER. */
    EXEC SQL ROLLBACK WORK;                /* Terminate whatever is outstanding */
    #ifdef CHECKOUT
        printf(" %ld", sqlca.sqlcode);    /* One normally expects 0 here, but */
    #endif
                                           /* occasionally something horrid has */
                                           /* been known to pop up here. */
    if (sqlca.sqlcode != 0) {
        MESSAGE;
        if ((sqlca.sqlcode != NOT_LOGGED_ON) &&
            (sqlca.sqlcode != NOT_CONNECTED)) {
            sprintf(workspace.arr, "Logbook choked on: %s",
                sqlca.sqlarm.sqlarmc);
            sprintf(dante, "\n%s %s\t\t\t making logbook entry for %s",
                clock(), sqlca.sqlarm.sqlarmc, ASSESSMENT.name);
            SLOCATE(workspace.arr);
        }
    }
    else { /* All is well and we are ready to make the logbook entry */
        username = ASSESSMENT.id;
        EXEC SQL SELECT TO_CHAR(SYSDATE, 'dd-moa-yy HH24:MI:SS')
            FROM DUAL INTO :timest2;
        EXEC SQL INSERT INTO register(planner, lastmtr, lastmoa,
            start_work, stop_work, idnumber)
            VALUES (:plannam, :lastmtr, :lastmoa,
                TO_DATE(:timest1, 'dd-moa-yy HH24:MI:SS'),
                TO_DATE(:timest2, 'dd-moa-yy HH24:MI:SS'),
                :username);
        #ifdef CHECKOUT
            printf("INSERT %ld", sqlca.sqlcode);
        #endif
        if (rooda = sqlca.sqlcode) { /* Just in case it fails.... */
            MESSAGE;
            sprintf(dante, "\n%s Logbook entry failed: %s",
                clock(), sqlca.sqlarm.sqlarmc);
            sprintf(dante, "\t\t\t %s (ID = %d)", ASSESSMENT.name, username);
            sprintf(dante, "\n%s\t\t\t %s, %s, %s",
                plannam.arr, timest1.arr, timest2.arr);
            EXEC SQL ROLLBACK WORK RELEASE;
            #ifdef CHECKOUT
                printf("Rolled back %ld", sqlca.sqlcode);
            #endif
        }
        else {
            EXEC SQL COMMIT WORK RELEASE;
            #ifdef CHECKOUT
                printf("Committed %ld", sqlca.sqlcode);
            #endif
            sprintf(dante, "\n%s Disconnected %s from ID = %d (%s)",
                clock(), plannam.arr, username, ASSESSMENT.name);
        }
        return rooda;
    }
}

else { /* If you are SUPERUSER, just make */
    EXEC SQL ROLLBACK WORK RELEASE;        /* a call to log off and return */
    #ifdef CHECKOUT
        printf(" %ld (SU Bypassed)", sqlca.sqlcode);
    #endif
}

return (int) sqlca.sqlcode;
}

```

```

int myName()
/*****
 *
 * myName -- To set the ORACLE USER information in the ASSESSMENT
 *           structure so it is available to all ASAN routines.
 *
 *           Looks in view SYS.V$KXUSER and retrieves USERID,
 *           MYNAME, TIME$TAMP and authorization codes.
 *
 * Returns the value of sqlca.sqlcode to calling program
 *
 *****/

```

```

{
#ifdef CHECKOUT
printf("\nmyname: ");
#endif

EXEC SQL SELECT myid, myprivs, myname
FROM sys.v$expuser
INFO :userno, :O_auth, :my_name;
my_name.arr[my_name.len] = '\0';

#ifdef CHECKOUT
printf(" %d - %d (%s)\n", sqlca.sqlcode, my_name.len, my_name.arr);
#endif
strcpy(ASSESSMENT.name, my_name.arr); /* Store results in global */
ASSESSMENT.id = userno; /* ASSESSMENT structure */
ASSESSMENT.auth[0] = O_auth.arr[0];
ASSESSMENT.auth[1] = O_auth.arr[1];
ASSESSMENT.auth[2] = O_auth.arr[2];

return (int) sqlca.sqlcode;
}

int enrollORA(name)
/*****
*
* enrollORA -- ASAN's SUPERUSER Grants CONNECT and RESOURCE
* to new ORACLE user
*
* Routine a). Signs on as SUPERUSER, if current user not DBA.
* b). Enrolls a new ORACLE USERNAME.
* c). Signs on under the new assessment's name.
*
* Returns to calling program with sqlca.sqlcode of last step executed.
* Routine terminates immediately if a sub-step returns a non-zero code.
*
* Returns (ZERO) if successful or (NON-ZERO) if not successful.
* Most likely, in order of appearance
* -0001 - Name already exists
* -0954 - No IDENTIFIED BY clause (i.e. blanks in name)
* -0967 - Illegal character in username
* -9999 - Any other ORACLE error that can occur
*
*****/
char name[];
{
int dlen, rcode, vfyOld(), SUCCESS();
register int i, j;
char *clock();
static char inputline[] = "entdesc ";

#ifdef CHECKOUT
printf("\n enrollORA");
#else
SLCOT("A moment, please, while ASAN updates its database.....");
#endif

if ( rcode = SUCCESS() ) { /* SUPERUSER's services are required to do this */
#ifdef CHECKOUT
printf(" %d", sqlca.sqlcode);
#endif
sprintf(workspace.arr, "SUPERUSER LOGIN REQUEST CRASHED Error %d.",
rcode);
SLCOTFW(workspace.arr);
SLCOTFW("You need serious help!.....");
sprintf(dante, "\n%s ENROLLORA: Superuser login crashed", clock());
ENDMSG;
sprintf(dante, "\n\t\t\t\t %s", sqlca.sqlwarn.sqlwarn);
closeORA();
exit(255);
}

j = n2bv.len = strlen(name); /* Convert name to uppercase because */
for (i = 0; i < j; i++) /* ORACLE keeps USERNAME in uppercase */
n2bv.arr[i] = name[i] = toupper(name[i]);
n2bv.arr[n2bv.len] = '\0';
strcpy(workspace.arr, "GRANT CONNECT, RESOURCE TO "); /* The GRANT statement */
strcat(workspace.arr, name); /* name is not part of */
strcat(workspace.arr, " IDENTIFIED BY "); /* ANSI standard SQL! */
strcat(workspace.arr, univpwd);
workspace.len = strlen(workspace.arr);

```

```

EXEC SQL EXECUTE IMMEDIATE :workspace;

#ifdef CHECKOUT
printf("\n%s %ld", workspace.arr, sqlca.sqlcode);
#endif

if (sqlca.sqlcode) { /* Who knows what evil lurks in the heart of ORACLE? */
    sprintf(dante, "\n%s Failed to enroll %s\n\t\t\t %s",
            clock(), name, sqlca.sqlarm.sqlarmec);
    return (int) sqlca.sqlcode; /* This traps any illegal characters */
                                /* and other strange things.... */
}

sprintf(dante, "\n%s Successfully enrolled %s", clock(), name);

EXEC SQL SELECT userid
      FROM sysuserlist
     INFO :uid
    WHERE username = :a2bv;

ADD_WINDOW( "newadesc" , 5, 5); /* User may enter up to 240 */
workspace.arr[0] = '\0'; /* characters of drivel to */
for (i = 0; i < 4; i++) { /* identify this assessment */
    inputline[7] = i+49;
    UPDATE DAFMS(inputline);
    entdesc[i].len = strlen(entdesc[i].arr);
    if (entdesc[i].len == 0) break;
    strcpy(workspace.arr, entdesc[i].arr);
    if (entdesc[i].len < 50) break;
}
workspace.len = strlen(workspace.arr);
REMOVE_WINDOW();

EXEC SQL INSERT INTO table_of_contents (idnumber, description)
      VALUES (:uid, :workspace);

if (sqlca.sqlcode) {
    #ifdef MSG;
    sprintf(dante, "\n%s Table of contents entry failed\n\t\t\t %s",
            clock(), sqlca.sqlarm.sqlarmec);
    SLOUTP(sqlca.sqlarm.sqlarmec);
    roode = (int) sqlca.sqlcode; /* This will cause the assessment */
                                /* to become invisible to ASAN or */
    EXEC SQL ROLLBACK WORK; /* may lead to security violation */
    #ifdef CHECKOUT
    printf("\n%s %ld", sqlca.sqlcode); /* errors that will stop execution */
    #endif
    }
else {
    EXEC SQL COMMIT WORK;
    roode = ASANroode(name); /* Once you get this far, sign on. */
    #ifdef CHECKOUT
    printf("\n%s %ld", sqlca.sqlcode); /* This better never be non-zero! */
    #endif
}
return roode;
}

int SUroode()
/*****
*
*      SUroode -- Connect ASAN's SUPERUSER to ORACLE
*
*      Returns SQLCA.SQLCODE
*
*****/
{
    static char dbaid[] = {0123,0125,0120,0105,0122,0125,0123,0105,0122,057,
                          0115,0105,0120,0110,0111,0123,0124,0117,0120,0110,
                          0105,0114,0105,0123,'\0'};

    int temp, logentry(), myname();

#ifdef CHECKOUT
printf("\n$SUroode: ");
#endif

logentry(); /* Make logbook entry and terminate whatever is outstanding */

strcpy( uid.arr, dbaid ); /* Connect */
uid.len = strlen( uid.arr );
EXEC SQL CONNECT :uid;

```

```

#ifdef CHECKOUT
printf(" %ld", sqlca.sqlcode);
#endif

if (!sqlca.sqlcode) myName();
strcpy(ASSESSMENT.desc, "Owner of ASAN's Administrative Information");
return (int) sqlca.sqlcode;
}

int vfyoid(name)
/*****
 *
 * vfyoid -- Verify that a USERNAME exists in ORACLE's Dictionary
 *
 * Determines if the username exists from the SYSUSERLIST View.
 * Returns SQLCA.SQLCODE of the SELECT statement.
 *
 *****/
char name[]; /* Argument is pointer to character string */
/* containing the username to validate */
{
register int i;

#ifdef CHECKOUT
printf("\nvfyoid");
#endif

a2bv.len = strlen(name); /* Set up for WHERE clause of the SELECT */
for (i = 0; i < a2bv.len; i++) /* ORACLE likes uppercase names */
a2bv.arr[i] = toupper(name[i]);

EXEC SQL SELECT userid
FROM sysuserlist
INFO :userid
WHERE username = :a2bv ;

#ifdef CHECKOUT
printf(" %ld", sqlca.sqlcode);
#endif

return (int) sqlca.sqlcode;
}

/*****
 *
 * housekpt.pc -- Set of routines to print "housekeeping" data from
 * the ORACLE Data Dictionary.
 *
 * Routines in this file:
 *
 * SUpriint - Allows SUPERUSER to print an assessments tables
 * SUpriopt - Selects level of detail for SUpriint or prints
 * HEADQUARTERS or SUPERUSER tables with that option
 * printab() - Prints names and comments on tables for assessment
 * printcol() - Prints names of all tables and names and comments
 * of columns within tables
 * printab() - Prints TABLE_OF_COMMENTS entries
 *
 *****/

#include <stdio.h> /* The usual stuff, of course */
#include <process.h> /* Header for calls to MS-DOS */
#include <string.h> /* String manipulation header */

#define SQLCA_STORAGE_CLASS extern
EXEC SQL INCLUDE SQLCA; /* SQL Communication Area */
EXEC SQL BEGIN DECLARE SECTION;
EXEC SQL INCLUDE hostvars.h;
EXEC SQL END DECLARE SECTION;
#include "asatype.h"
#include "asan.h" /* Standard ASAN Header file */

static printoption = 0;

int SUpriint(name)
/*****
 *

```

```

*      SUpri() --- Print tables for an assessment (called from the
*      ===== housekeeping screen while SUPERUSER is connected
*
*      Routine logs on as the named assessment, prints tables as that
*      assessment, returns to SUPERUSER, puts up housekeeping screen again.
*
*      =====/
char name [];
{
char *clock();

strcpy (uid.arr, name);
uid.len = strlen(uid.arr);
strcpy (pwd.arr, uid.pwd);
pwd.len = strlen(pwd.arr);
EXEC SQL ROLLBACK WORK RELEASE;
EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
if (sqlca.sqlcode) {
    SENDMSG;
    sprintf(dante, "\n%s Could not connect to %s for printing tables",
            clock(), name);
    sprintf(dante, "\n\t\t\t %s", sqlca.sqlarm.sqlarmc);
    SLOUTP("Could not connect to assessment");
    SLOUTP(sqlca.sqlarm.sqlarmc);
}
else {
    myname();
    if (printoption == 2) prntool();
    else prntab();
}
strcpy(ASSESSMENT.name, "SUPERUSER");
SUconn();
if (sqlca.sqlcode) {
    SENDMSG;
    sprintf(dante, "\n%s Could not reconnect after printing tables for %s",
            clock(), name);
    sprintf(dante, "\n\t\t\t %s", sqlca.sqlarm.sqlarmc);
    SLOUTP("Could not reconnect to SUPERUSER");
    SLOUTP(sqlca.sqlarm.sqlarmc);
    exit(255);
}
NEW_SCREEN("dbhousekpscreen");
return (int) 0;
}

```

```

int SUpript(value)
/*****
*
*      SUpript() --- Set options for printing assessments or print the
*      ===== HEADQUARTERS or SUPERUSER tables
*
*      Routine logs on as the named assessment, prints tables as that
*      assessment, returns to SUPERUSER, puts up housekeeping screen again.
*
*      =====/
int value;
{
char *clock();
#ifdef CHECKROOT
    printf("\nSUpript %d", value);
#endif

switch (value) {
case 1:
    printoption = 1;
    if (ulisto()) {
        SENDMSG;
        SLOUTP(sqlca.sqlarm.sqlarmc);
        return (int) -1;
    }
    blankdpl();
    NEW_SCREEN("elcasscreen");
    ununch();
    return (int) 0;
case 2:
    printoption = 2;
    if (ulisto()) {
        SENDMSG;
        SLOUTP(sqlca.sqlarm.sqlarmc);
    }
}
}

```

```

        return (int) -1;
    blankdopl();
    NEW_SCREEN("slmsscreeen");
    ununch();
    return (int) 0;

case 3:
    ulists();
    optrl = hqocaa;
    EXEC SQL ROLLBACK WORK RELEASE;
    EXEC SQL CONNECT :optrl;
    if (sqlca.sqlcode) {
        MSGMSG;
        sprintf(dante, "\n%s Could not connect to HEADQUARTERS for printing tables",
            clock());
        sprintf(dante, "\n\t\t\t %s", sqlca.sqlwarn.sqlwarn);
        SLOUTP("Could not connect HEADQUARTERS");
        SLOUTP(sqlca.sqlwarn.sqlwarn);
    }
    else {
        myOcama();
        if (printoption == 2) prntool();
        else prntabs();
    }
    strcpy(ASSESSMENT.name, "SUPERUSER");
    SUCocaa();
    if (sqlca.sqlcode) {
        MSGMSG;
        sprintf(dante, "\n%s Could not reconnect after printing HEADQUARTERS tables",
            clock());
        sprintf(dante, "\n\t\t\t %s", sqlca.sqlwarn.sqlwarn);
        SLOUTP("Could not reconnect to SUPERUSER");
        SLOUTP(sqlca.sqlwarn.sqlwarn);
        exit(255);
    }
    NEW_SCREEN("dbhsckpgscreen");
    return (int) 0;

default:
    ulists();
    if (printoption == 2) prntool();
    else prntabs();
    }
    NEW_SCREEN("dbhsckpgscreen");
    return (int) 0;
}

```

```

int prntabs()
/*****
*
*   prntabs()   ---   Print a list of all tables for this assessment
*
*
*   Routine opens cursor U2 then fetches rows until SQL_EOF is found.
*   Each row printed on the system printer. Then the cursor is closed.
*
*   Note:  Modifications to this function may impact the related
*          functions tlisto(), tlistf() and tlists() that open the
*          cursor, fetch rows using it respectively, and close it
*
*****/
{
#define PAGESIZE 54
#undef COMMENTSSPACE
#define COMMENTSSPACE 54
int i, j, line, lcp, nsp, page;          /* Counters for lines, etc. */

int tlisto();                          /* Database Utilities Used */
int tlistf();
int tlists();

SLOUTP("Printing tables");
if ( ! tlisto() ) {                    /* Open sys.viampstab query */
    j = 0;
    page = 1;
    sprintf( prn, "\t\tInventory of Tables for %s\t\tPage %d\n",
        ASSESSMENT.name, page);
    line = 2;

    while ( sqlca.sqlcode != SQL_EOF ) {

```



```

if ( tlistf() == SQL_EOF ) break;      /* Get name of the next table */
if ( line >= PAGESIZE ) {              /* Make sure that you have space */
    sprintf( prn, "\f\t\tInventory of Tables for %s\t\tPage %d\n",
        ASSESSMENT.name, ++page);
    line = 2; }

j++;                                  /* Print name of the table */
sprintf( prn, "\n%d\t\t-20s ", j, tid.arr); /* Assumes <= 20 char! */

lop = 0; /* Print as many characters as the comment field contains */
while (lop < workspace.len) {
    ncp = (lop+COMMENTSPACE < workspace.len) ?
        lop+COMMENTSPACE : workspace.len;
    for ( ; ncp > lop; ncp--) /* Find good spot to end this line */
        if (workspace.arr[ncp] == ' ' ||
            workspace.arr[ncp] == '\0') break;

    if (lop == ncp) { /* In case we have a line without white space */
        sprintf( prn, " ");
        for ( ; ncp < lop+COMMENTSPACE; ncp++)
            sprintf( prn, "%c", workspace.arr[ncp]);
        sprintf( prn, "\n\t\t\t ");
        line++; }

    else { /* The normal case when we can find a white space somewhere */
        workspace.arr[ncp] = '\0';
        sprintf( prn, "%s\n\t\t\t ", &workspace.arr[lop]);
        line++;
        lop = ncp+1; }
    line++;
} /* End WHILE (lop) */
}. /* End WHILE (sqlca) */
sprintf( prn, "\n\n\t\t\t\t== END ==\f");
tlistf();
}

#ifdef CHECKOUT
else {
    printf("\n\nBOUCH! My cursor did not open!");
    expMsg();
}
#endif
}

int prntool()
/*****
*
* prntool() --- Print a list of all columns CREATED BY this
*              assessment sorted by table in which they occur
*
* Routine opens cursor U4 then fetches rows until SQL_EOF is found.
* Each row printed on the system printer. Then the cursor is closed.
*
* Note: Modifications to this function may impact the related
*       functions tlistto(), tlistf() and tlistc() that open the
*       cursor, fetch rows using it respectively, and close it
*
*****/
{
#define PAGESIZE 54
#undef COMMENTSPACE
#define COMMENTSPACE 40

int tlistto(); /* Database Utilities Used */
int tlistf();
int tlistc();

int i, j, k, line, lop, ncp, page; /* Counters for lines, etc. */
char qual_col[31];

SLOCUT("Printing tables and columns");
if ( ! tlistto() ) { /* Open "COL" DataDict Query */
    j = k = 0;
    page = 1;
    sprintf( prn, "\f\t\tInventory of Columns by Table for %s\t\tPage %d",
        ASSESSMENT.name, page);
    line = 3;
    strcpy(qual_col, "q");

    while ( sqlca.sqlcode != SQL_EOF ) {

```



```
#ifndef CHECKOUT
    printf("\aQUAL_CIT1 tld", sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT ) {
    ENDMSG;
    SLOUTP(sqlca.sqlarmc.sqlarmc);
    Sprintf(dante, "\a%s %s\a\t\t\t Creating Table QUAL_CIT1 (ID = %d)",
        clock(), sqlca.sqlarmc.sqlarmc, ASSESSMENT.id);
    return (int) sqlca.sqlcode;}

EXEC SQL CREATE TABLE QUAL_CIT2(entry_num char(5));

#ifdef CHECKOUT
    printf("\aQUAL_CIT2 tld", sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT ) {
    ENDMSG;
    SLOUTP(sqlca.sqlarmc.sqlarmc);
    Sprintf(dante, "\a%s %s\a\t\t\t Creating Table QUAL_CIT2 (ID = %d)",
        clock(), sqlca.sqlarmc.sqlarmc, ASSESSMENT.id);
    return (int) sqlca.sqlcode;}

return (int) sqlca.sqlcode;
}
```

```
#include <stdio.h>

#define SQLCL_STORAGE_CLASS extern      /* Switch for header files      */
#include SQL_INCLUDE SQLCL;
```



```

EXEC SQL BEGIN DECLARE SECTION;      /* All SQL declarations are in */
EXEC SQL INCLUDE hostvars.h;        /* these header files          */
EXEC SQL INCLUDE faharris.h;

EXEC SQL END DECLARE SECTION;
#include "asaa.h"                    /* Standard ASAA Header File */

static missflg = 0;

int pchgmis()
/*****
 *
 *      pchgmis  -- Set up routine to select a new mission
 *
 *
 *      Routine (1) opens cursor for fetch of all missions known (in this
 *                  version missions are always local to the assessment)
 *      (2) fetches the first batch into memory
 *      (3) puts up "chgcurstrscreen"
 *
 *****/
{
    int blankdpl(), sllsto(), subunch(), roode;

#ifdef CHECKOUT
    printf("\npchgmis ");
#endif

    WHEREAMI(Screen, Window, Button, Datum); /* Save where you were when you */
    stropy(oldscren, Screen);                /* called this "pseudo pop-up" */

    blankdpl();
    NEW_SCREEN("chgmis");
    SLOUT("Retrieving list of missions");
    roode = sllsto();
    return (roode ? roode : subunch());
}

int pentrmis()
/*****
 *
 *      pentrmis  -- Set up routine to select a mission and
 *                  aircraft combination for a given MTR
 *
 *
 *      Routine (1) sets mission flag to false
 *      (2) puts up "spentrmis"
 *
 *****/
{
    missflg = 0;
    stropy(tid.arr, ac_name.arr);
    tid.len = ac_name.len;
    NEW_SCREEN("spentrmis");
    return (int) 0;
}

int entarmis()
/*****
 *
 *      entarmis  -- Enter a mission for this assessment
 *
 *
 *****/
{
    int roode;
    void expMsg();
    char *clock();

#ifdef CHECKOUT
    printf("\nentarmis ");
#endif

    if (ac_in_form <= 0) {
        SLOUTSP("? Sortie size");
        UPDATE_DATUM("numac");
    }

    misstype = toupper(misstype);

```

```

missdesc.len = strlen(missdesc.arr);

EXEC SQL INSERT INTO missions (mission, type, descr, sortia_size)
VALUES (:misslabl, :misstype, :missdesc, :ac_in_form);
#endif CHECKOUT
printf(" %ld", sqlca.sqlcode);
#endif

if (rcode = (int) sqlca.sqlcode) {
    expmsg();
    fprintf(dante, "\n%s entameis: %s", clock(), sqlca.sqlarm.sqlarmc);
    EXEC SQL ROLLBACK WORK;
    missflg = 0;
} else {
    EXEC SQL COMMIT WORK;
    missflg = 1;
}
NEW_SCREEN(oldscreen);
NEWVALS();
return rcode;
}

int psmtrflt()
/*****
*
*      psmtrflt -- Set up routine to process a mission and
*      aircraft combination for a given MTR
*
*****/
{
    int cancmis(); solisto(), solistf(), solista(), vfyacmtr();
    void expmsg();
    char *clock();

    if (vfyacmtr(ac_name.arr) /* Do we know how to calculate this? */
        return (int) sqlca.sqlcode;

    if (!missflg) { /* Has a mission been entered? */
        SLOUTP("Please Select Mission First");
        return (int) SQL_EOF;
    }

#endif CHECKOUT
    printf("\nPrevious? %s %s %s", srcid.arr, misslabl.arr, ac_name.arr);
#endif

EXEC SQL SELECT activity FROM activities INTO :activity
WHERE S_LABEL = :srcid
AND M_IDENT = :misslabl
AND AIRCRAFT = :ac_name;

#endif CHECKOUT
printf(" = %ld (code = %ld)", activity, sqlca.sqlcode);
SLOUTP("ack");
#endif

if (sqlca.sqlcode == SQL_EOF) { /* This is a unique combination */

    EXEC SQL SELECT MAX(activity) FROM activities INTO :activity;
#endif CHECKOUT
    printf("\nLast activity = %ld code = %ld", activity, sqlca.sqlcode);
#endif
    if (sqlca.sqlcode)
        if (sqlca.sqlcode == SQL_EOF) activity = 0;
        else {
            expmsg();
            fprintf(dante, "\n%s Psmtrflt2: %s", clock(), sqlca.sqlarm.sqlarmc);
            return (int) sqlca.sqlcode;
        }
    activity++;

EXEC SQL INSERT INTO activities (S_LABEL, M_IDENT, AIRCRAFT, ACTIVITY)
VALUES (:srcid, :misslabl, :ac_name, :activity);
#endif CHECKOUT
printf(" Insert = %ld", sqlca.sqlcode);
SLOUTP("ack");
#endif
if (sqlca.sqlcode) {
    expmsg();
    fprintf(dante, "\n%s Psmtrflt3: %s", clock(), sqlca.sqlarm.sqlarmc);
    return (int) sqlca.sqlcode;
}
ournavpt.arr[ournavpt.len = 0] = '\0';
prenavpt.arr[prenavpt.len = 0] = '\0';

```

```

    lptr1 = &carlowalt.altitude;
    cptr1 = &carlowalt.units;
    NEWVALS();
    NEW_SCREEN("strflt");
}
else {
    if (sqlca.sqlcode) {
        expMsg();
        sprintf(dante, "\n%s Pstrflt1: %s", clock(), sqlca.sqlerrm.sqlerrmc);
        cancmis();
        return (int) sqlca.sqlcode;
    }
    soliste();
    if (sqlca.sqlcode) {
        expMsg();
        sprintf(workspace.arr, "FAILED - %s", sqlca.sqlerrm.sqlerrmc);
        sprintf(dante, "\n%s OPEN %s", clock(), workspace.arr);
        sprintf(dante, "\n%s\t\t Retrieving Operations for activity %ld",
            activity);
        cancmis();
        return (int) sqlca.sqlcode;
    }
    for (;;) {
        solistf();
        if (sqlca.sqlcode) {
            if (sqlca.sqlcode != SQL_EOF) {
                expMsg();
                sprintf(workspace.arr, "FAILED - %s", sqlca.sqlerrm.sqlerrmc);
                sprintf(dante, "\n%s SELECT %s", clock(), workspace.arr);
                sprintf(dante, "\n%s\t\t Retrieving Month %d Operations for %ld",
                    active_mo, activity);
                cancmis();
            }
            break;
        }
        ops[active_mo-1].day = ops_day;
        ops[active_mo-1].nite = ops_nite;
    }
    if (sqlca.sqlcode == SQL_EOF) {
        soliste();
        EXEC SQL DELETE FROM operations WHERE activity = :activity;
        #ifdef CHECKOUT
            printf("\nPrevious data deleted %ld", sqlca.sqlcode);
            SLOUTP("ack");
        #endif
    }
    else soliste();
    ADD_WINDOW("month", 7, 1);
    SLOUTP("You already have data!");
}
return (int) 0;
}

```

```

int penumis(name)
/*****
*
*      penumis  -- Set up routine for entering a new mission
*
*
*      Routine (1) verifies that the new name is unique.
*      (2) puts up the mission definition screen and starts it
*
*****/

```

```

char name[];
{
    char *clock();
    int roode, meliste(), mabunch();
    void expMsg();

    #ifdef CHECKOUT
        printf("\npenumis ");
    #endif

    REMOVE_WINDOW();
    SLOUT("Processing New Mission Request");

    if ((roode = vfythis(name)) != SQL_EOF) { /* Check for strange things */
        if (roode == 0) {
            sprintf(workspace.arr, "Sorry, but %s already exists", name);
            SLOUTP(workspace.arr);
        }
    }
}

```

```

        return (-1);}
expMsg();
sprintf(dante, "\n%s PName: %s", clock(), sqlca.sqlarm.sqlarmc);
return rodc;
}
else {
    selists(); /* Close list of choices */
    strcpy (misslabl.arr, a2bv.arr); /* This is the new mission */
    misslabl.len = a2bv.len;
    misdesc.len = 0;
    misdesc.arr[0] = '\0';
    NEW_SCREEN("misdesc");
    NEWVALS();
    /* NEW_PARAM("misdesc"); Removed for possible conflict with U Bag */
    return (int) 0;
}
}

```

```

int actmispt()
/*****
 *
 *   actmispt -- Store an MFR Navigation Point actually flown on a
 *   _____ particular activity and advance all pointers.
 *
 *****/
{
    register int i;
    int inamispt();

#ifdef CHECKOUT
    printf("\nactmispt ");
#endif

    if (!inamispt()){
        /* Advance the Nav Point Parameters */

        for (i = 0; i < 3; i++) prenavpt.arr[i] = curnavpt.arr[i];
        for (i = 0; i < 10; i++) prelowalt.spec[i] = curlowalt.spec[i];
        ac_pre_spd = ac_cur_spd;
        ac_pre_pwr = ac_cur_pwr;

        curnavpt.arr[0] = '\0';
        NEWVALS();
        NEW_PARAM("curnavpt1");
    }
    return (int) sqlca.sqlcode;
}

```

```

int inamispt()
/*****
 *
 *   inamispt -- Insert an MFR Navigation Point actually traversed
 *   _____ during a particular mission in the database
 *
 *****/
{
    register int i;
    void expMsg();

#ifdef CHECKOUT
    printf("\ninamispt ");
#endif

    strcpy(a2bv.arr, staid.arr);
    strcpy(a2bv.arr, curnavpt.arr);
    a2bv.len = strlen(a2bv.arr);
    fltseq += 1;

    EXEC SQL INSERT INTO MFR FLIGHT PARAM
        (ACTIVITY,    FIX_LABEL,    ALT_REF,    ALT,
         PWR,         SPD,          SEQ)
    VALUES(
        :activity,    :a2bv,        :cptrl,    :lptrl,
        :ac_cur_pwr,  :ac_cur_spd,  :fltseq);

#ifdef CHECKOUT
    printf(" Insert = %ld", sqlca.sqlcode);

```

```

#endif
if (sqlca.sqlcode) {
    if (sqlca.sqlcode == EXISTS) SLOUTP("Duplicate Entry");
    else expmsg();
    cncmis(); /* This is necessary since ORACLE automatically rolls back */
              /* the inserts already done at this point ! */
}

return (int) sqlca.sqlcode;
}

int psaddops(count)
/*****
 *
 * psaddops -- Put up the add operations window and accept data.
 *
 *****/
{
    int count;
    {
        if (count == 1) ADD_WINDOW("daynite", 6, 1);
        else if (count == 12) ADD_WINDOW("month", 6, 1);
        else SLOUTP("PSADDOPS called with bad argument");
        return (int) 0;
    }
}

int expmops()
/*****
 *
 * expmops -- Expand steady yearly operations over all months.
 *
 *****/
{
    register int i;

    for (i = 1; i < 12; i++) {
        ops[i].day = ops[0].day;
        ops[i].nite = ops[0].nite;
    }
    return (int) 0;
}

int vfyMsis(name)
/*****
 *
 * vfyMsis -- Verify the existence of a mission as part
 *           of an assessment
 *
 * Routine looks in the assessment's MISSIONS table, loads mission
 * into :cid and returns sqlca.sqlcode for the query.
 *****/
{
    char name[];
    {
        register int i;
        #ifdef CHECKOUT
        printf("\nvfyMsis ");
        #endif

        n2bv.len = strlen(name);
        for (i = 0; i < n2bv.len; i++)
            n2bv.arr[i] = toupper(name[i]);
        n2bv.arr[i] = '\0';

        EXEC SQL SELECT mission
            FROM missions
            INTO :cid
            WHERE mission = :n2bv;

        cid.arr[cid.len] = '\0';
        #ifdef CHECKOUT
        printf(" %ld ", sqlca.sqlcode);
        #endif
    }
}

```

```

return (int) sqlca.sqlcode;
}

```

```

int cancelis()
/*****
 *
 *      cancelis  --  Cancel the mission currently pending on the database
 *
 *
 *      Routine always issues a ROLLBACK
 *
 *****/
{
    int pextrant();

#ifdef CHECKOUT
    printf("\ncancelis ");
#endif

    EXEC SQL ROLLBACK WORK;
#ifdef CHECKOUT
    printf("tld", sqlca.sqlcode);
#endif
    if (
        sprintf(workspace.arr, "Entry for mission %s CANCELLED", misslabl.arr);
        SLOUTMS(workspace.arr);
        misslabl.arr[misslabl.len - 1] = '\0';
        pextrant();
        return (int) sqlca.sqlcode;
    )
}

```

```

int saveamis()
/*****
 *
 *      saveamis  --  Commit the mission currently pending on the database
 *
 *
 *
 *****/
{
    int mrtabl(), pextrant();

#ifdef CHECKOUT
    printf("\nsaveamis ");
#endif

    for (active_no = 1; active_no < 13; active_no++) {
        ops_day = ops[active_no-1].day;
        ops_nite = ops[active_no-1].nite;
        EXEC SQL INSERT INTO operations ( ACTIVITY, MONTH, DAY, NIGHT, LASTUD)
            VALUES (:activity, :active_no, :ops_day, :ops_nite, SYSDATE);
        if (sqlca.sqlcode) {
            MSGMSG;
            sprintf(workspace.arr, "FAILED - %s", sqlca.sqlwarn.sqlwarnm);
            sprintf(dante, "\n%s INSERT %s", clock(), workspace.arr);
            sprintf(dante, "\n\t\t\t\t Storing Operations %d tld",
                active_no, activity);
            EXEC SQL ROLLBACK WORK;
            pextrant();
            return (int) sqlca.sqlcode;
        }
    }

    EXEC SQL COMMIT WORK;

    mrt_tabl();

    pextrant();

    return (int) sqlca.sqlcode;
}

```

```

int MISocan(name)
/*****
 *
 *
 *****/

```

```

*      MISscon -- Connect an MFR with a Mission
*
*
*      Routine SELECTS the Mission from the database and loads parameter
*      block. Returns an error code which indicates whether or not
*      the SELECT was successful.
*
*      Note: The existence of the mission on the database is assumed.
*      SQLCA.SQLCODE is returned and if nonzero an error
*      message is displayed on the status line.
*
*****/
char name[];
{
  register int i;
  int NEW_SCREEN(), NEWVALS(), SLOUT();
  char *clock();

#ifdef CHECKOUT
  printf("\nMISscon ");
#endif

  SLOUT("Retrieving Mission");
  n2bv.len = strlen(name);
  for (i = n2bv.len; i >= 0; i--) n2bv.arr[i] = name[i] = toupper(name[i]);

  EXEC SQL SELECT mission, type, descr, sortie_size
    FROM   missions
    DFO    :mislabl, :misstype, :misdscr, :as_in_form
    WHERE  mission = :n2bv;

#ifdef CHECKOUT
  printf("%ld ", sqlca.sqlcode);
#endif

  if (!sqlca.sqlcode) { /* Everything is O.K. */
    mislabl.arr[mislabl.len] = '\0';
    misdscr.arr[misdscr.len] = '\0';
    slisto();
    missflg = 1;
    NEWVALS();
    NEW_SCREEN(oldscreens);
  }
  else { /* This should never happen! But,..... */
    missflg = 0;
    NEWMSG;
    sprintf(workspace.arr, "FAILED - %s", sqlca.sqlerrm.sqlerrmc);
    fprintf(dante, "\n%s SELECT %s", clock(), workspace.arr);
    fprintf(dante, "\n\t\t Retrieving Mission %s", n2bv.arr);
    return (int) sqlca.sqlcode;
  }
}

int slisto()
/*****
*
*      slisto --- Open cursor S1 for a list of missions that
*      belong to the current assessment
*
*      Routine executes an open cursor command for cursor S1 and then
*      returns to the calling program with the ORACLE status code.
*
*      Note: Modifications to this function may impact the related
*      functions slistf(), sabunch() and slistc() that fetch
*      rows and close the cursor and, possibly, functions that
*      call these utility routines.
*
*****/
{
#ifdef CHECKOUT
  printf("\nslisto ");
#endif

  EXEC SQL DECLARE S1 CURSOR FOR SELECT mission
    FROM missions ORDER BY mission;

  EXEC SQL OPEN S1;

#ifdef CHECKOUT
  printf(" Open: %ld ", sqlca.sqlcode);

```

```

#endif
return (int) sqlca.sqlcode;
}

```

```

int slistf()
/*****
 *
 * slistf --- Fetch a row using the opened cursor S1 for missions
 *
 *
 * Routine executes an fetch command for cursor S1, which is assumed
 * to have been opened, and then returns to the calling program with
 * the ORACLE status code.
 *
 * Note: Modifications to this function may impact the related
 * functions slisto(), sdbunch() and slisto() that open,
 * fetch groupwise and close the cursor and, likely,
 * functions that call these utility programs
 *
 *****/
{
EXEC SQL FETCH S1 INTO :misslabl;
misslabl.arr[misslabl.len] = '\0';
#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

```

```

int sdbunch()
/*****
 *
 * sdbunch --- Fetch a bunch (20 or whatever the size of deplmult)
 * using the opened cursor S1 for mission list
 *
 *
 * Routine executes an fetch command for cursor S1, which is assumed
 * to have been opened, and then returns to the calling program with
 * the ORACLE status code.
 *
 * Note: Modifications to this function may impact the related
 * functions slisto(), slistf() and slisto() that open,
 * fetch and close the cursor and, likely, functions that
 * call these utility programs
 *
 *****/
{
register int i;
int nrows, nrowc, rowc;
char *clock();

EXEC SQL FETCH S1 INTO :deplmult;
nrows = (int) sqlca.sqlarrd[2];
nrowc = (sizeof deplmult) / 34;
rowc = (int) sqlca.sqlcode;

#ifdef CHECKOUT
printf(" sdbunch: %ld returns %d of %d rows", sqlca.sqlcode, nrowc, nrowc);
#endif
if ((rowc == SQL_FETCH_OUT_OF_ORDER) || (rowc == SQL_EOF)) {
if ((rowc == SQL_EOF) && (nrowc > 0)) {
for (i = 0; i < nrowc; i++) deplmult[i].arr[deplmult[i].len] = '\0';
if (nrowc < nrowc)
for (i = nrowc; i < nrowc; i++) deplmult[i].arr[0] = '\0';
NEWVALS();
SLOUTP("The last mission in the list is on the screen");
}
else {
NEWVALS();
SLOUTP("You are already as far down in the list as you can go");
}
}
else {
if (rowc) {
#ifdef MSG
fprintf(dante, "\n%s MESSAGE: %s", clock(), sqlca.sqlarrm.sqlarrmc);
NEWVALS();
}
return rowc;
}
}
}

```



```

int solista()
/*****
*
*      solista  ---  Close cursor S1 for mission list
*      -----
*
*      Routine executes a close cursor command for cursor S1 and then
*      returns to the calling program with the ORACLE status code.
*
*      Note:  Modifications to this function may impact the related
*             functions solisto(), solunch, and solistf() that open the
*             cursor and fetch rows using it and, possibly, functions
*             that call these utilities
*
*****/
{
#ifdef CHECKOUT
printf("\nsolista ");
#endif

EXEC SQL CLOSE S1;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

```

```

int solisto()
/*****
*
*      solisto  ---  Open cursor S2 for a list of operations that
*      -----      belong to the current activity
*
*      Routine executes an open cursor command for cursor S2 and then
*      returns to the calling program with the ORACLE status code.
*
*      Note:  Modifications to this function may impact the related
*             functions solistf() and solista() that fetch rows and
*             close the cursor and, possibly, functions that call
*             these utility routines.
*
*****/
{
#ifdef CHECKOUT
printf("\nsolisto ");
#endif

EXEC SQL DECLARE S2 CURSOR FOR SELECT month, day, night, lastupd
FROM operations WHERE activity = :activity ORDER BY month;

EXEC SQL OPEN S2;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

```

```

int solistf()
/*****
*
*      solistf  ---  Fetch a row using the opened cursor S2 for operations
*      -----
*
*      Routine executes an fetch command for cursor S2, which is assumed
*      to have been opened, and then returns to the calling program with
*      the ORACLE status code.
*
*      Note:  Modifications to this function may impact the related
*             functions solisto() and solista() that open and close
*             the cursor and, likely, functions that call these utilities
*
*****/
{
EXEC SQL FETCH S2 INTO :active_no, :ops_day, :ops_nite, :timeast2;

```

```

timet2.arr[timet2.lca] = '\0';
#ifdef CHECKOUT
printf(" solistc: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int solistc()
/*****
*
*      solistc  ---  Close cursor S2 for operations list
*      =====
*
*      Routine executes a close cursor command for cursor S2 and then
*      returns to the calling program with the ORACLE status code.
*
*      Note:  Modifications to this function may impact the related
*            functions solistf() and solistf() that open and fetch
*            rows from the cursor and, possibly, functions that call
*            these utilities
*
*****/
{
#ifdef CHECKOUT
printf("\solistc ");
#endif

EXEC SQL CLOSE S2;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}
/*****
*
*      mtr.pc  --  Routines calculating noise exposure from MTRs.
*      =====
*
*      This file contains:
*
*      isperpen  -  Return distance of point to line if perpendicular is
*                  inside line segment, NULL otherwise
*      g_mtrseal -  Get the 2-parameter SEAL distance approximation for
*                  an aircraft
*      mtr_seal  -  Calculate the Single Event Exposure Level for an MTR
*                  segment.
*      mtr_tabl  -  Calculate the noise/distance table for an MTR segment
*                  when the computational parameters have been loaded
*      MTRocaa   -  Connect an assessment to an MTR
*
*****/
#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>
#include <math.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION; /* All SQL declarations are in */
EXEC SQL INCLUDE hostvars.h;    /* these header files */
EXEC SQL INCLUDE faharris.h;

VARCHAR rowid[25];

EXEC SQL END DECLARE SECTION;
#include "asan.h"              /* Standard ASAN Header File */

double delta_pwr;             /* Global: adjustment for actual power setting */
double delta_spd;             /* Global: adjustment for actual aircraft speed */
double XO, YTO;               /* Global: coordinates of point of observation */
double XI, FYI;               /* Global: coordinates of a segment end point */
double XZ, YZ;                /* Global: coordinates of a segment end point */

double isperpen()
/*****
*
*      isperpen  --  Routine to determine if the perpendicular of
*
*****/

```

```

*           point (X0,Y0) to a line segment (X1,Y1)-(X2,Y2) *
*           falls inside the line segment. *
* * * * *
* Routine performs a translation of the coordinate system so that *
* (X0,Y0) is at the origin of the new coordinate system. It then *
* rotates this new coordinate system around (X0,Y0) to a second *
* new coordinate system in which the X-axis is parallel to the line *
* segment. If the perpendicular to the line segment is inside the *
* segment, then in this coordinate system the X-coordinates of the *
* end points of the segment will have different signs. In this *
* system, the absolute value of the Y-coordinate of either point is *
* the distance from the point to the segment. *
* * * * *
* Notes:  1. If the Y' value (in the second coordinate system) *
*           is positive, a person moving from (X1,Y1) to (X2,Y2) *
*           will observe (X0,Y0) as being to his/her right. *
* * * * *
*           2. The original coordinates are (X1,Y1), etc. *
*           The translated coordinates are (Xlp,Ylp), etc. *
*           The translated and rotated coordinates are (Xldp,Yldp) *
*           etc. to mimic the usual notational convention of *
*           primed and double-primed coordinate pairs. *
* * * * *
*****/

```

```

{

double Xp0, Xlp, X2p, Yp0, Ylp, Y2p;
double Xdp0, Xldp, X2dp, Ydp0, Yldp, Y2dp;

double sqrt();
double hypotenuse, sintheta, cosintheta;

#ifdef CHECKOUT
printf("\nisperpen ");
#endif

hypotenuse = sqrt( (X2 - X1)*(X2 - X1) + (Y2 - Y1)*(Y2 - Y1) );
sintheta   = (X2 - X1) / hypotenuse;
cosintheta = (Y2 - Y1) / hypotenuse;

Xlp = X1 - X0;
Ylp = Y1 - Y0;
X2p = X2 - X0;
Y2p = Y2 - Y0;

Xldp = Xlp * cosintheta + Ylp * sintheta;
Yldp = -Xlp * sintheta + Ylp * cosintheta;
X2dp = X2p * cosintheta + Y2p * sintheta;
Y2dp = -X2p * sintheta + Y2p * cosintheta;

#ifdef CHECKOUT
printf("\nsla = %7.5f Cos = %7.5f Syp = %6.2f",
       sintheta, cosintheta, hypotenuse);
printf("\nX0 = (%5.1f,%5.1f)", X0, Y0);
printf("\nX1 = (%5.1f,%5.1f) Y = (%5.1f,%5.1f)", X1, Y1, X2, Y2);
printf("\nX' = (%5.1f,%5.1f) Y' = (%5.1f,%5.1f)", Xlp, Ylp, X2p, Y2p);
printf("\nX'' = (%5.1f,%5.1f) Y'' = (%5.1f,%5.1f)", Xldp, Yldp, X2dp, Y2dp);
#endif
return ( ( (Xldp > 0) && (X2dp > 0) ) ||
         ( (Xldp < 0) && (X2dp < 0) )
         ? TRUE
         : false(Y2dp) );
}

```

```

int g_mtrsel()
/*****
*
*   g_mtrsel -- Routine to get the MTR two-parameter approximation
*               to the SHL - Distance curve from the database
*
* Routine selects the proper profile from the database and loads
* the host variables. It returns the sqlca.sqlcode of the select.
*
* Note: An error will result if the entries in the database are
*       not unique! At this time no provision is made for a
*       pick-and-choose approach to SHL profiles.
*
*****/

```

```

*****/
{
int temp;
#ifdef CHECKOUT
printf("\nag_mtr_seal ");
#endif
EXEC SQL SELECT power_units, power, power_scale,
slope, intercept, speed
INTO :pr_pwr_u, :pr_power, :pwr_scale,
:pwr_slope, :pwr_intcpt, :pr_speed
FROM mtrsaltab
WHERE aircraft = :ac_name;
#ifdef CHECKOUT
sprintf(prn, "\nats %1f %s speed = %1f, sqlcode %1d\n",
ac_name.arr, pr_power, pr_pwr_u.arr, pr_speed, sqlca.sqlcode);
printf("%1d", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

```

```

double mtr_seal(lateral)
/*****
*
* mtr_seal -- Routine to calculate noise exposure at a point that
* is lateral to an MTR for a SINGLE specific activity
* on that MTR. (Operational adjustments not made.)
*
*****/

```

```

double lateral; /* Lateral distance to track */
{
double log_angle; /* LOG10(angle above the horizon) */
double onset; /* Rise time parameter */
double onset_pen; /* Penalty assoc'd therewith */
double saladj; /* SEL adjusted for power and speed */
double atan2(), log10(), pow(); /* math.h routines used */

```

/* Calculate "Pure" Propagation-adjusted SEL */

```

#ifdef CHECKOUT
printf("\nAltitude %1f Lateral %1f", ac_alt, lateral);
sprintf(prn, "\nAltitude %5.01f Lateral %5.01f", ac_alt, lateral);
sprintf(prn, " intercept %7.31f slope %7.31f delta P %7.31f S %7.31f",
pwr_intcpt, pwr_slope, delta_pwr, delta_spd);
#endif

```

log_angle = 1.750122632 + log10(atan2(ac_alt, lateral));

```

saladj = pwr_intcpt
+ pwr_slope * ( 0.5 * log10 (ac_alt*ac_alt + lateral*lateral) )
+ delta_pwr /* Power Adjustment */
+ delta_spd /* Speed Adjustment */
- (log_angle < 1.477121255 ? /* Ground adjustment if < 30 deg */
(6.995 - 6.606 * log_angle
+ 1.566 * log_angle * log_angle) : 0.0);

```

/* Calculate Rise Time Penalty */

```

onset = 100.0 / (1+ pow(2.7183, 10.01 - 3.62 * log10(ac_speed)
+ 2.48 * log10(ac_alt)
+ 0.15 * log10(lateral)
- 0.0542 * saladj));

```

```

if (onset < 15.0) onset_pen = 0.0;
else if (onset <= 30.0) onset_pen = 16.6 * log10(onset/15.0);
else onset_pen = 5.0;

```

```

#ifdef CHECKOUT
sprintf(prn, "saladj %10.41f onset %10.41f", saladj, onset);
#endif
return (saladj + onset_pen);
}

```

int mtr_tabl()

```

/*****
 *
 *      mtr tabl  -- Routine to calculate noise exposure - perpendicular
 *                  distance table (MTR_EXP_TAB) for a specific activity
 *
 *****/
*****
***  NOTE:  This routine currently assumes that operations parameters ***
***         are fixed over an MTR.  It must be changed when that is ***
***         no longer true.  Also INFLIE.PC must then be changed! ***
*****
/*****
 *
 *      char *clock();
 *      int  alistf(), alistf(), alistf(), i;
 *      int  g_mtrsel();          /* Finds profile for this aircraft */
 *      double mtr_sel();         /* Single event calculation routine */
 *      void expMsg();

SLOUT("Calculating MTR Exposure Table");
strcpy(tid.arr, "MTR_EXP_TAB");
tid.len = 11;
sprintf(a2bv.arr, "ACT%04ld", activity);
a2bv.len = strlen(a2bv.arr); /* Just in case activity > 9999 */

/*****
 *
 *      This query only computes things for the 1st segment.  If we start
 *      allowing different operations parameters per segment, then this
 *      must be changed.  Also see the note in INFLIE.PC!
 *
 *****/

EXEC SQL SELECT alt,      pwr,      spd      FROM mtr_flight_param
      INFO :ac_alt, :ac_power, :ac_speed
      WHERE activity = :activity AND seq = 1;

#ifdef CHECKOUT
      printf("\nAircraft is %ld, %ld KTS, power = %ld sqlcode = %ld",
             ac_alt, ac_speed, ac_power, sqlca.sqlcode);
      sprintf(prn, "\nAircraft is %ld, %ld KTS, power = %ld sqlcode = %ld",
             ac_alt, ac_speed, ac_power, sqlca.sqlcode);
#endif

alistf(); /* Open database cursor */
for (;;) {
      alistf(); /* Fetch the next instance */
      if (!sqlca.sqlcode) {
             cid.arr[cid.len] = '\0';
             if (!strcmp(cid.arr, a2bv.arr)) break; /* Have it already! */
             else cid.arr[0] = '\0'; cid.len = 0; /* No, not that one */
             else if (sqlca.sqlcode == SQL_EOF) break; /* Have to create it */
             else { /* Trouble! */
                     expMsg();
                     sprintf(dante, "\n%s %s", clock(), sqlca.sqlrm.sqlrmno);
                     fprintf(dante, "\n\t\t\t Retrieving MTR_EXP_TAB %s", a2bv.arr);
                     SLOUTS("Calculation Aborted");
                     closeORA();
                     return (int) sqlca.sqlcode;
             }
      }
}
alistf(); /* Close database cursor */

if (!cid.len) { /* Create a new column in the table */
      sprintf(workspace.arr, "ALTER TABLE MTR_EXP_TAB ADD (%s NUMBER)", a2bv.arr);
      workspace.len = strlen(workspace.arr);
      #ifdef CHECKOUT
             printf("\n%s", workspace.arr);
             fprintf(prn, "\n%s", workspace.arr);
      #endif
      EXEC SQL EXECUTE IMMEDIATE :workspace;
      #ifdef CHECKOUT
             printf("\nAlter Table %s %ld", a2bv.arr, sqlca.sqlcode);
             fprintf(prn, "\nAlter Table %s %ld", a2bv.arr, sqlca.sqlcode);
      #endif
      if (sqlca.sqlcode) {
             expMsg();
             sprintf(dante, "\n%s %s", clock(), sqlca.sqlrm.sqlrmno);
             fprintf(dante, "\n\t\t\t Creating MTR_EXP_TAB %s", a2bv.arr);
             SLOUTS("Calculation Aborted");
      }
}

```

```

        closeORA();
        return (int) sqlca.sqlcode;
    }

    sprintf(workspace.arr,
        "COMMENT ON COLUMN MTR_EXP_TAB.ts IS 'Operations on ts by ts on mission ts'",
        a2bv.arr, aroid.arr, aq_name.arr, misslabl.arr);
    workspace.len = strlen(workspace.arr);
    #ifdef CHECKOUT
        printf("\nts", workspace.arr);
    #endif
    EXEC SQL EXECUTE IMMEDIATE :workspace;
}

g_strreal();
delta_pwr = (a2_pwr - pr_pwr) * pwr_scale;
delta_spd = 10.0 * log10(pr_spd / a2_spd);
sprintf(workspace.arr,
    "SELECT rowid, sideline, ts FROM mtr_exp_tab FOR UPDATE OF ts",
    a2bv.arr, a2bv.arr);
workspace.len = strlen(workspace.arr);
#ifdef CHECKOUT
    printf("\naprepare ts", workspace.arr);
#endif

EXEC SQL PREPARE D1 FROM :workspace;
#ifdef CHECKOUT
    printf("\na returns %ld", sqlca.sqlcode);
    sprintf(prn, "\nts returns %ld", workspace.arr, sqlca.sqlcode);
#endif
EXEC SQL DECLARE C1 CURSOR FOR D1;
EXEC SQL OPEN C1;
#ifdef CHECKOUT
    printf("\nopen returns %ld", sqlca.sqlcode);
    sprintf(prn, "\nopen returns %ld", sqlca.sqlcode);
#endif

for (;;) {
    EXEC SQL FETCH C1 INTO :rowid, :sideline, :exposure;
    #ifdef CHECKOUT
        printf("\nfetch sideline %ld returns %ld", sideline, sqlca.sqlcode);
        sprintf(prn,
            "\nfetch sideline %ld returns %ld", sideline, sqlca.sqlcode);
    #endif
    if ((sqlca.sqlcode) && (sqlca.sqlcode != NULL_FETCHED)) break;
    exposure = mtr_seal(sideline);
    if (rowid.len != 18) {
        sprintf(prn, "\nrowid length = %d", rowid.len);
        rowid.len = 18;
    }
    sprintf(workspace.arr,
        "UPDATE MTR_EXP_TAB SET ts = %ld WHERE rowid = '%s'",
        a2bv.arr, exposure, rowid.arr);
    workspace.len = strlen(workspace.arr);
    EXEC SQL EXECUTE IMMEDIATE :workspace;
    #ifdef CHECKOUT
        printf("\nts RETURNS %ld", workspace.arr, sqlca.sqlcode);
        sprintf(prn, "\nEXEC IMM ts RETURNS %ld", workspace.arr, sqlca.sqlcode);
    #endif
}
if (sqlca.sqlcode == SQL_EOF) { /* We finished the list */
    EXEC SQL COMMIT WORK;
}
else {
    expMsg();
    sprintf(dante, "\nts %s", clock(), sqlca.sqlwarn.sqlwarn);
    sprintf(dante, "\n\t\t\t Fetching MTR_EXP_TAB for activity %ld %ld",
        activity, sideline);
    EXEC SQL ROLLBACK WORK;
}
EXEC SQL CLOSE C1;
}

```

```

int MTRconn(name)
/*****
 *
 *      MTRconn -- Connect an ASAN assessment with an MTR
 *
 *
 *
 */

```

```

* Routine SELECTS the MFR from the database and loads parameter *
* block. Returns an error code which indicates whether or not *
* the SELECT was successful. *
* *
* Note: The existence of the MFR on the database is assumed. If *
* any SQL error is found, the routine returns sqlca.sqlcode *
* with an error on the status line. When the MFR exists *
* the MFR Data Entry Screen will be displayed. To verify *
* the existence of an MFR use vfyMfr(). *
* *
*****/
char name[];
{
register int i;
int NEW_SCREEN(), NEWVALS(), SLOUT(), SLOUTSP();
char *clock();

#ifdef CHECKROUT
printf("\nMFRoooo ");
#endif

SLOUT("Switching MFRs ....");
a2bv.len = strlen(name);
for (i = a2bv.len; i >= 0; i--) a2bv.arr[i] = name[i] = toupper(name[i]);
EXEC SQL SELECT label, status, type, descr, orig, sched, owner,
TO_CHAR(date_pab, 'dd-Mon-YYYY'),
TO_CHAR(timestamp, 'dd-Mon-yy HH4:MI:SS')
FROM sources
INFO :sroid, :srostat, :srotype, :srodesc, :sroorig, :srosched,
:sroowner, :srodate, :srodate
WHERE label = :a2bv AND type = '1';

#ifdef CHECKROUT
printf("%ld ", sqlca.sqlcode);
#endif

if (!sqlca.sqlcode) { /* Everything is O.K. */
sroid.arr[sroid.len] = '\0';
srodesc.arr[srodesc.len] = '\0';
sroorig.arr[sroorig.len] = '\0';
srosched.arr[srosched.len] = '\0';
strcpy(lastmfr.arr, sroid.arr);
lastmfr.len = sroid.len;
melistc();
mfrtrist();
NEW_SCREEN("mfrtry");
NEWVALS();
if ((srostat != 'A')) {
sprintf(workspace.arr, "Note: Status of this MFR is '%c'", srostat);
SLOUTSP(workspace.arr);
}
} else { /* This should never happen! But,..... */
MESSAGE:
sprintf(workspace.arr, "FAILED - %s", sqlca.sqlwarn.sqlwarn);
if (sqlca.sqlcode != SQL_EOF) SLOUTSP(workspace.arr);
else SLOUTSP("This MFR exists but is not available to you");
sprintf(dante, "\n%s SELECT %s", clock(), workspace.arr);
sprintf(dante, "\n\t\t\t Retrieving MFR %s", a2bv.arr);
return (int) sqlca.sqlcode;
}
}
/*****
*
* screensa.pc -- Routines to set up the screens for assessments.
*
* This file contains:
*
* loggedon - Answers the question: "Did the planner log on?"
* pedbheak - Puts up database housekeeping screen
* pedchpas - Puts up change assessment screen
* penwases - Puts up new assessment screen
* peprobet - Puts up problem status screen
* blinkdepl - Blanks out the common display area
*
*****/
#include <process.h> /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA STORAGE_CLASS extern /* Switch for header files */
EXEC SQL INCLUDE SQLCA;

```

```

EXEC SQL BEGIN DECLARE SECTION;      /* All SQL declarations are in */
EXEC SQL INCLUDE hostvars.h;        /* these header files          */
EXEC SQL INCLUDE sbarris.h;
EXEC SQL END DECLARE SECTION;
#include "asan.h"                     /* Standard ASAN Header File  */

int loggedon()
/*****
 *
 *   loggedon  -- Routine to determine if the planner filled in the
 *               username on the first screen.
 *
 *****/
{
#define CHAR_HERO '0'

char *clock();
int i, j, closeORA();

#ifdef CHECKOUT
printf("\nloggedon ");
#endif

WHEREAMI(Screen, Window, Datum, Button);

#ifdef CHECKOUT
printf("called from %s", Screen);
#endif

planname.len = strlen(planname.arr);      /* Who is this person? */
for (i = 0; i < planname.len; i++) {
    if (planname.arr[i] > CHAR_HERO) break;
    planname.len--;
    for (j = 0; j < planname.len; j++) planname.arr[j] = planname.arr[j+1];
}

#ifdef CHECKOUT
printf(" Username = %s is %d characters", planname.arr, planname.len);
#endif
if (planname.len == 0) { /* Did not enter a name! (or we lost it) */
    if (strcmp(Screen, "firstscreen")) {
        SLOCUS("TROUBLE! ASAN has forgotten your name.....");
        sprintf(dante, "\n%s Loggedon: ASAN \"lost\" planner's name", clock());
        sprintf(dante, "\n%s\t\t\t ASSESSEST.name was %s", ASSESSEST.name);
        sprintf(dante, "\n%s\t\t\t n2bv.arr was %s", n2bv.arr);
        sprintf(dante, "\n%s\t\t\t Screen %s", Screen);
        sprintf(dante, "\n%s\t\t\t Window %s", Window);
        sprintf(dante, "\n%s\t\t\t Datum %s", Datum);
        sprintf(dante, "\n%s\t\t\t Button %s", Button);
        NEW_SCREEN("firstscreen");
    }
    /*
    closeORA();
    exit(255);
    */
    return (int) 255;
}
else {
    SLOCUS("I don't know yet who you are. Please enter your name");
    UPDATE_DATUM("planname");
    VCAPITAL(&planname);
    ADD_WINDOW("password", 15, 3);
    UPDATE_DATUM("password");
    VCAPITAL(&password);
    REMOVE_WINDOW();
}
return (int) SQL_EOF;
}

return (int) 0;
}

int pedbhsak()
/*****
 *
 *   pedbhsak  -- Set up routine for housekeeping screen
 *
 *   Routine (1) makes sure that user is known
 *
 *****/

```



```

*          (2) makes an entry in the audit file          *
*          (3) puts up "dbhsakpgscreen"                  *
*                                                                 *
*****/
int closeORA(), loggedon();
char *clock();

loggedon();
fprintf(dante,
        "\n\n%s PDBRSEK: %s signed on as SUPERUSER for File Maintenance\n\n",
        clock(), planname.art);
if (strcmp(ASSESSMENT.name, "SUPERUSER") ) {
    if (SUOcan()) {
        fprintf(dante, "%s PDBRSEK: SUPERUSER connect failed\n\n", clock());
        fprintf(dante, "\n\t\t\t %s", sqlca.sqlwarn.sqlwarn);
        SLOUTP("SUPERUSER access denied. Not good.....");
        closeORA();
        exit(256);
    }
}
NEW_SCREEN("dbhsakpgscreen");
return 0;
}

```

```

pchgass()
/*****
*          pchgass -- Set up routine for change assessment screen
*          ****
*
*          Routine (1) opens cursor for fetch of ORACLE usernames
*          (2) fetches the first batch into memory
*          (3) puts up "chgcurassscreen"
*
*****/
{
    int blankdepl(), ulisto(), ubunch(), roode;

#ifdef CHECKOUT
    printf("\napschgass ");
#endif

    blankdepl();
    NEW_SCREEN("chgcurassscreen");
    SLOUT("Retrieving ASAN's table of contents");
    roode = ulisto();
    if (!roode) roode = ubunch();
    return roode;
}

```

```

int penvasen(name)
/*****
*          penvasen -- Set up routine for new assessment screen
*          ****
*
*          Routine (1) verifies that the new name is unique.
*          (2) creates the new ORACLE user and tables for this
*              assessment.
*
*          Notes: 1. When routine determines that it is stuck, closeORA
*                  is called to terminate the execution.
*                2. Execution is terminated when this routine is called
*                  without being connected to ORACLE. (One should not
*                  be able to get oneself into this predicament.)
*
*****/
char name[];
{
    char *clock();
    int roode, ASANocan(), intise(), arollORA(), SUOcan();
    int ulisto(), ubunch(), ulisto();
    void expOmsg();

#ifdef CHECKOUT

```

```

printf("\npsnwasen ");
#endif

REMOVE_WINDOW();
SLOUTP("Processing New Assessment Request");

if (( roode = vfyOld(name)) != SQL_EOF) { /* Check for strange things */
    if ( roode == NOT_LOGGED_ON || roode == SQL_BAD_LOGIN) {
        sprintf(workspace.arr,
            "PSnwasen: You are not connected to ORACLE! -- FATAL");
        SLOUTP(workspace.arr);
        sprintf(dante, "\n%s %s", clock(), workspace.arr);
        exit(255);
    }
    if ( roode == 0) {
        sprintf(workspace.arr, "Sorry, but %s already exists", name);
        SLOUTP(workspace.arr);
        return (-1);
    }
    expMsg();
    return roode;
}
else { /* This is where you land when everything is O.K. */
    ulisto(); /* Close the list of choices */
    roode = arollORA(name); /* Try to get this one added */
    if ( roode ) {
        sprintf(workspace.arr, "FAILED -- %s", sqlca.sqlarm.sqlarmc);
        sprintf(dante, "\n%s %s FAILED to enroll", clock(), name);
        sprintf(dante, "\n\t\t\t\t\t %s", sqlca.sqlarm.sqlarmc);
        if (SUCOAN()) {
            sprintf(workspace.arr, "PSNwasen stack: %s!",
                sqlca.sqlarm.sqlarmc);
            SLOUTP(workspace.arr);
            sprintf(dante, "\n%sPSNwasen stack: SUPERUSER re-connect failed",
                clock());
            sprintf(dante, "\n\t\t\t\t\t %s", sqlca.sqlarm.sqlarmc);
            closeORA();
            exit(255);
        }
        if (!ulisto()) { /* Recover by reinitializing screen */
            ununch();
            NEWVALS();
        }
        if ((roode == SQL_BAD_LOGIN) || (roode == NO_IDENTIFIED_BY)) {
            sprintf(workspace.arr,
                "FAILED -- Are there perhaps spaces in \"%s\"?", name);
            SLOUTP(workspace.arr);
        }
    }
    else { /* arollORA() executed O.K. Next */
        if (!initize()) return 0; /* create the ORACLE tables needed */
        #ifdef CHECKOUT
            printf("\nSomething major is wrong. Code = %d", roode);
        #endif
        SLOUTP("Initialization terminated abnormally");
        SLOUTP(sqlca.sqlarm.sqlarmc);
        SLOUTP("This should be fixed before entering data");
        sprintf(dante, "\n%s Initialization did not complete normally", clock());
        sprintf(dante, " due to\n\t\t\t\t\t %s", sqlca.sqlarm.sqlarmc);
        sprintf(dante, "\n\t\t\t\t\t This problem should be fixed before proceeding");
        sprintf(dante, " with\n\t\t\t\t\t data entry for %s", name);
    }
}
}

```

```

int pprobet()
/*****
 *
 * pprobet -- Routine to set up the AMAN Status Screen. This
 *          displays the ASSESSMENT structure. If SUPERUSER
 *          is the current user, the system loads the last
 *          active assessment first.
 *
 *****/
{
    int closeORA(), lastsess(), loggedon(), roode, statusflag;
    char *clock();

    #ifdef CHECKOUT
        printf("\npprobet ");
    #endif
    loggedon();
    if (!strcmp(ASSESSMENT.name, "SUPERUSER")) { /* Not CONNECTED ? */
        statusflag = lastsess(); /* Get last session */
        if (statusflag == 0) { /* Found it! */

```

```

EXEC SQL SELECT username          /* Find the name of the */
FROM sysuserlist                 /* last assessment */
INFO :a2bv                       /* worked on */
WHERE userid = :username;

#ifdef CHECKOUT
printf(" %ld", sqlca.sqlcode);
#endif
if (sqlca.sqlcode == SQL_EOF) { /* Trouble in River City... */
    SLOUTB("PSPROBST detected logbook violation. ID = %d", username);
    sprintf(dante, "\n%s PSPROBST detected logbook violation. ID = %d",
        clock(), username);
    closeORA();
    exit(255);
}
rooda = ASAMocan(a2bv.arr); /* Sign on as that last assessment */

if (rooda) {
    SLOUTB("ASAMocan error not trapped");
    HEDCMG;
    SLOUTB(sqlca.sqlwarn.sqlwarnm);
}
return rooda; /* ASAMocan will have put up the proper screen */
}

else {
    if (statusflag == SQL_EOF) {
        SLOUTB("No work over done yet: You can only start a new one");
        blankdpl();
        NEW_SCREEN("ahgcurasscreen");
        return 0;
    }
    expMsg();
    return statusflag;
}
}

else { /* When you are already connected to ASAM as a regular user */
    /* You YOURSELF are now the last user and the time is NOW! */

    strcpy(planr1st.arr, planrnm.arr);
    EXEC SQL SELECT TO_CHAR(SYSDATE, 'dd-Mon-yy HH24:MI:SS')
    FROM dual into :lastdate;
    NEWVALS();
    NEW_SCREEN("probetatscreen");
    return 0;
}
}

int blankdpl()
/*****
*
* blankdpl -- Routine to blank out the common display area.
* This is a safety precaution, since if the list
* to be displayed is empty, whatever was left from
* the previous list would show. This is because
* the "bunch" routines don't "remember" how often
* they have been called and so don't blank out a
* display when they run out of data!
*
*****/
{
    register int m, i;
    m = (sizeof dplmalt) / 34;
    for (i = 0; i < m; i++) {
        dplmalt[i].arr[0] = '\0';
        dplmalt[i].len = 0;
    }
    NEWVALS();
}

/*****
*
* screenm.pc -- Routines to set up the screens for MTRs.
*
* This file contains:
*
* postrent - Puts up screen to work with MTRs
* pschgstr - Puts up screen to select a new MTR
* psnmstrn - Puts up screen to define a new MTR
* psmlsreq - Puts up screen to define mission requirements
* mltmrtxt - Make a textblock of an MTR's Navigation Point user data
* entmtrpt - Starts the MTR entry process
*
*****/

```



```

        return (int) sqlca.sqlcode;
    }
    else {
        srodesc.arr[srodesc.lca] = '\0';
        sroedate.arr[sroedate.lca] = '\0';
        sropdate.arr[sropdate.lca] = '\0';
        srosched.arr[srosched.lca] = '\0';
        sroorig.arr[sroorig.lca] = '\0';
        sroid.arr[sroid.lca] = '\0';
        NEWVALS();
        NEW_SCREEN("ntentry");

        return (int) sqlca.sqlcode;
    }
}

```

```

pschgptr()
/*****
 *
 *      pchgptr  -- Set up routine for screen to change to new MFR
 *
 *
 *      Routine (1) opens cursor for fetch of all MFR names known to system
 *      (2) fetches the first batch into memory
 *      (3) puts up "chgcurstrscreen"
 *
 *****/
{
    int blankdpl(), mlistc(), mabunch(), roode;

#ifdef CHECKOUT
    printf("\npschgptr ");
#endif

    blankdpl();
    NEW_SCREEN("chgcurstrscreen");
    SLOC("Retrieving ASAM's list of accessible MFRs");
    roode = mlistc();
    return (roode ? roode : mabunch());
}

```

```

int penumtra(name)
/*****
 *
 *      penumtra  -- Set up routine for new MFR entry screen
 *
 *
 *      Routine (1) verifies that the new name is unique. If not, the
 *      reason (whether it is already on your list or another
 *      assessment already has exclusive use of it) is shown
 *      (2) puts up the MFR definition screen and starts it
 *
 *****/
{
    char name[];
    {
        char *alock();
        int roode, mlistc(), mabunch();
        void expMsg();

#ifdef CHECKOUT
        printf("\npenumtra ");
#endif

        REMOVE_WINDOW();
        SLOC("Processing New MFR Request");

        if ( ( roode = vfyMfr(name) ) != SQL_EOF ) { /* Check for strange things */
            if ( roode == 0 ) {
                sprintf(workspace.arr, "sorry, but %s already exists", name);
                SLOC(workspace.arr);
                return (-1);
            }
            if ( roode == DUPLICATE_OBJECT ) {
                EXEC SQL SELECT username

```

```

FROM sysuserlist
INFO :tid
WHERE userid = :sroouser;
tid.arr[tid.lem] = '\0';
sprintf(workspace.arr,
  "Sorry! %s already \"owns\" %s", tid.arr, name);
SLOUTES(workspace.arr);
return (-1);}

ampMsg();
fprintf(dante, "\n%s %s\n", clock(), sqlca.sqlwarn.sqlwarn);
crostat = 'U';
return roode;}

else {
  mlistc(); /* Close list of choices */
  strcpy(srocid.arr, n2bv.arr); /* This is now the mtr name */
  srocid.lem = n2bv.lem;
  EXEC SQL SELECT TO_CHAR(SYSDATE, 'dd-mon-yyyy')
  FROM dual INTO :srodate; /* Suggest a date of Publ. */
  srodate.arr[srodate.lem] = '\0';
  #ifdef CHECKOUT
    printf("\nDATE (sqlcode = %ld) %s %s", sqlca.sqlcode,
      srodate.lem, srodate.arr);
  #endif
  srodesc.lem = 0;
  srodesc.arr[0] = '\0';
  crostat = 'A';
  NEW_SCREEN("mtrdefinescreen"); /* Put up data entry screen */
  NEWVALS();
  return (int) 0; }
}

```

```

int pemisreq()
/*****
 *
 * pemisreq -- Routine to set up the Mission Requirements Screen.
 *
 *****/
{
  int NEW_SCREEN();

  #ifdef CHECKOUT
    printf("\npemisreq ");
  #endif

  NEW_SCREEN("misreq");
  return 0;
}

```

```

int maktrtxt()
/*****
 *
 * maktrtxt -- Create a textblock of an MTR's Navigation Points
 *            user information (Identifier, Fix info, etc.)
 *
 *****/
{
  char *clock();
  FILE *fopen();

  int i, j, mlistc(), mlistf(), mlists(), fclose();

  #ifdef CHECKOUT
    printf("\nmaktrtxt ");
  #endif
  if ((txtblkf = fopen("txtblk\mtr.txt", "w")) == NULL) {
    SLOUTES("Error opening file for help window");
    fprintf(dante, "\n%s could not open MTR textblock file for %d",
      clock(), ASSESMENT.id);
    return (int) DUPLICATE_OBJECT;}

  fprintf(txtblkf, "  Navigation Points for %s are:\n\n", srocid.arr);
  mlistc();
  if (!sqlca.sqlcode) {
    for (i = 0; i < n2bv.lem; i++) {
      mlistf();
      if (sqlca.sqlcode) break;
      for (j = 0; j < n2bv.lem; j++)

```



```

ent.latitude      = OFF_ERROR;
ent.longitude     = OFF_ERROR;
ent.fixrad        = 0 ;
ent.fixdist       = 0 ;

INTERVALS();
}
return (int) sqlca.sqlcode;
}

int inamtrpt()
/*****
 *
 *      inamtrpt -- Insert an MFR Navigation Point into the database
 *
 *
 *****/
{
    int canamtr();
    register int i;
    void expMsg();

#ifdef CHECKOUT
    printf("\namamtrpt ");
#endif

    if (COORD2utm(ent)) return (int) -1;

    strcpy(n2bv.arr, srcid.arr);
    strcpy(n2bv.arr, curnavpt.arr);
    n2bv.laa = strlen(n2bv.arr);

EXEC SQL INSERT INTO MFRNAVPOINTS
        (FIX_LABEL, FLOOR_REF, CEILING_REF, FIX_ID,
         FIX_TYPE, ARFCO, FIX_LAT, FIX_LONG,
         FIX_RAD, FIX_DIST, FLOOR, CEILING,
         WIDTH_LEFT, WIDTH_RIGHT)
VALUES (:n2bv, :dptr1, :dptr2, :curfixid,
        :curfixtyp, :curarctco, :dptr3, :dptr4,
        :curfixrad, :curfixdist, :lptr1, :lptr2,
        :curwidleft, :curwidright);

#ifdef CHECKOUT
    printf(" Insert1 = %ld", sqlca.sqlcode);
#endif
if (sqlca.sqlcode) {
    if (sqlca.sqlcode == EXISTS) SLOUTERR("Duplicate Navpoint Id");
    else expMsg();
    canamtr(); /* This is necessary since ORACLE automatically rolls back */
               /* the inserts already done at this point ! */
} else {
EXEC SQL INSERT INTO NAVPOINTS ( FIX_LABEL, X, Y, LAT, LONG )
VALUES (:n2bv, :dptr1, :dptr2, :dptr3, :dptr4);

#ifdef CHECKOUT
    printf(" Insert2 = %ld", sqlca.sqlcode);
#endif
if (sqlca.sqlcode) {
    expMsg();
    if (sqlca.sqlwarn[0]) expWarn();
    canamtr(); /* ORACLE automatically rolls back */
               /* inserts after this occurs ! */
}
return (int) sqlca.sqlcode;
}

int vfyamtr(name)
/*****
 *
 *      vfyamtr -- Verify the existence of aircraft data for
 *
 *      MFR calculations in the database
 *
 *
 *
 *      Routine looks in MFRSEL/AS (a view on a HEADQUARTERS table), loads
 *      pr_pwr_u with the power units for which we have data for the aircraft
 *      and returns sqlca.sqlcode for the query. Zero means MFR exists,
 *      SQL_EOF means it does not, all other values indicate an SQL error.
 *
 *****/

```

```

*
*****/

char name[];
{
    register int i;
    char *clock();

#ifdef CHECKOUT
    printf("\navfyacstr ");
#endif

    a2bv.len = strlen(name);
    for ( i=0; i < a2bv.len; i++)
        a2bv.arr[i] = name[i] = toupper(name[i]);
    pr_pwr_u.arr[0] = '\0';
    pr_pwr_u.len = 0;

EXEC SQL SELECT power_units          /* See if aircraft power units */
FROM mtrsalstab                     /* are in the list of aircraft */
INFO :pr_pwr_u                      /* for which we have MTR data */
WHERE aircraft = :a2bv;

#ifdef CHECKOUT
    printf(" %ld %s power in %s", sqlca.sqlcode, a2bv.arr, pr_pwr_u.arr);
#endif

    if (sqlca.sqlcode) {
        if (sqlca.sqlcode == SQL_EOF) {
            sprintf(workspace.arr, "MTR Calculation not supported for %s", a2bv.arr);
            SLOUTAB(workspace.arr);
        }
        else {
            MESSAGE;
            sprintf(date,
                "%ats %s/%s/%s looking for %s in MTRSKI/TAB for %d",
                clock(), sqlca.sqlwarn.sqlwarno, a2bv.arr, ASSESSMENT.Id);
        }
    }
    else {
        pr_pwr_u.arr[pr_pwr_u.len] = '\0';
        for ( i=0; i < a2bv.len; i++)
            ac_name.arr[i] = a2bv.arr[i];
        ac_name.len = i;
        ac_name.arr[i] = '\0';
        EXEC SQL SELECT power, speed    /* Load the reference conditions */
FROM mtrsalstab                     /* as initialization for this */
INFO :ac_cur_pwr, :ac_cur_spd        /* Aircraft/Mission */
WHERE aircraft = :ac_name;
        ac_pre_pwr = ac_cur_pwr;
        ac_pre_spd = ac_cur_spd;
#ifdef CHECKOUT
        printf(" %ld %s power in %s", sqlca.sqlcode, a2bv.arr, pr_pwr_u.arr);
#endif

        NEWVALS(); }
    return (int) sqlca.sqlcode;
}

int vfyMtr(name)
/*****
*
*      vfyMtr  --  Verify the existence of a MTR on the system
*
*
*      Routine looks in SOURCELIST (a view on a SUPERUSER table), loads
*      srownum and returns sqlca.sqlcode for the query.  Zero means MTR
*      exists, SQL_EOF means it does not, all other values indicate an SQL
*      error except that DUPLICATE_OBJECT means that the object exists on
*      the system but is not accessible to the present assessment
*
*****/

char name[];
{
    register int i;
#ifdef CHECKOUT
    printf("\navfyMtr ");
#endif

```

```

a2bv.len = strlen(name);
for ( i=0; i < a2bv.len; i++)
    a2bv.arr[i] = toupper(name[i]);

EXEC SQL SELECT owner          /* Identify the owner from */
FROM    sourcelist            /* the list of all sources */
INFO    :srowowner
WHERE    label = :a2bv         /* which has this name and */
AND type = '1';              /* is an MFR */

#ifdef CHECKOUT
    printf(" %ld owner %d", sqlca.sqlcode, srowowner);
#endif

if (sqlca.sqlcode == NULL_FETCHED) return (int) 0;
if ((sqlca.sqlcode == 0) && (srowowner != ASSESSMENT.id))
    return (int) DUPLICATE_OBJECT;
return (int) sqlca.sqlcode;
}

int canceltr()
/*****
 *
 *      canceltr -- Cancel the MFR currently pending on the database
 *
 *      Routine always issues a ROLLBACK
 *
 *****/
{
    int posttrans();

#ifdef CHECKOUT
    printf("\ncanceltr ");
#endif

EXEC SQL ROLLBACK WORK;
#ifdef CHECKOUT
    printf("%ld", sqlca.sqlcode);
#endif
    sprintf(workspace.arr, "Entry for MFR %s CANCELLED", srcid.arr);
    SLOUTS(workspace.arr);
    srcid.arr[srcid.len - 1] = '\0';
    posttrans();
    return (int) sqlca.sqlcode;
}

int savetr()
/*****
 *
 *      savetr -- Commit the MFR currently pending on the database
 *
 *      Routine checks if there is a valid MFR being constructed then
 *      issues a COMMIT to the database
 *
 *****/
{
    register int i;

#ifdef CHECKOUT
    printf("\nsavetr ");
#endif

    mllists();
    if (srcstat == '0') {
        SLOUTS("MFR had errors and was not saved");
        EXEC SQL ROLLBACK WORK;
#ifdef CHECKOUT
            printf("ROLLED MFR BACK!..... %ld", sqlca.sqlcode);
#endif
        srcid.arr[srcid.len - 1] = '\0';
    }
    else { /* First check if we have to enter one more point! */
        if (ournavpt.arr[0] != '\0')

```

```

        if (isnstrpt()) return (int) sqlca.sqlcode;

EXEC SQL COMMIT WORK;
#ifdef CHECKOUT
    printf("Committed MFR %ld", sqlca.sqlcode);
#endif
    }
    postrent();
    return (int) sqlca.sqlcode;
}

```

```

int mlistc()
/*****
 *
 *      mlistc  ---  Open cursor S1 for a list of MFRs on the system
 *      -----          that are accessible to an assessment
 *
 *      Routine executes an open cursor command for cursor S1 and then
 *      returns to the calling program with the ORACLE status code.
 *
 *      Note:  Modifications to this function may impact the related
 *             functions mlistf(), mabunch() and mlistc() that fetch
 *             rows and close the cursor and, possibly, functions that
 *             call these utility routines.
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nmlistc ");
#endif

EXEC SQL DECLARE S1 CURSOR FOR
    SELECT label          /* These are unique identifiers */
    FROM   sources        /* View of all accessible "sources" */
    WHERE  type = '1'     /* MFRs are type == 1 */
    ORDER BY label;

EXEC SQL OPEN S1;

#ifdef CHECKOUT
    printf(" Open: %ld ", sqlca.sqlcode);
#endif
    return (int) sqlca.sqlcode;
}

```

```

int mlistf()
/*****
 *
 *      mlistf  ---  Fetch a row using the opened cursor S1 for MFRs
 *      -----
 *
 *      Routine executes an fetch command for cursor S1, which is assumed
 *      to have been opened, and then returns to the calling program with
 *      the ORACLE status code.
 *
 *      Note:  Modifications to this function may impact the related
 *             functions mlistc(), mabunch() and mlistc() that open,
 *             fetch groupwise and close the cursor and, likely,
 *             functions that call these utility programs
 *
 *****/
{
EXEC SQL FETCH S1 INTO :sroid;
sroid.arr[sroid.len] = '\0';
#ifdef CHECKOUT
    printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
    return (int) sqlca.sqlcode;
}

```

```

int mabunch()
/*****
 *
 *      mabunch  ---  Fetch a bunch (20 or whatever the size of default)
 *      -----          using the opened cursor S1 for mtrlist
 *

```

```

*
* Routine executes an fetch command for cursor S1, which is assumed
* to have been opened, and then returns to the calling program with
* the ORACLE status code.
*
* Note: Modifications to this function may impact the related
* functions mlisto(), mlistf() and mlistc() that open,
* fetch and close the cursor and, likely, functions that
* call these utility programs
*
*****/
{
register int i;
int rows, nrow, rows;
char *clock();

EXEC SQL FETCH S1 INTO :deplmult;
rows = (int) sqlca.sqlerrd[2];
nrow = (sizeof deplmult) / 34;
rows = (int) sqlca.sqlcode;

#ifdef CHECKOUT
printf(" mlistc: %ld returns %d of %d rows", sqlca.sqlcode, rows, nrow);
#endif
if ((rows == SQL_FETCH_OUT_OF_ORDER) || (rows == SQL_EOF)) {
if ((rows == SQL_EOF) && (rows > 0)) {
for (i = 0; i < rows; i++) deplmult[i].arr[deplmult[i].len] = '\0';
if (rows < nrow)
for (i = rows; i < nrow; i++) deplmult[i].arr[0] = '\0';
NEWVALS();
SLOUTP("The last MFR in the list is on the screen"); }
else {
NEWVALS();
SLOUTP("You are already as far down in the list as you can go"); }
}
else {
if (rows) {
NEWMSG;
sprintf(dante, "%s MESSAGE: %s", clock(), sqlca.sqlerrm.sqlerrmc);
NEWVALS();}
return rows;
}

int mlistc()
/*****
*
* mlistc --- Close cursor S1 for mlist
*
* Routine executes a close cursor command for cursor S1 and then
* returns to the calling program with the ORACLE status code.
*
* Note: Modifications to this function may impact the related
* functions mlisto(), mlistf(), and mlistc() that open the
* cursor and fetch rows using it and, possibly, functions
* that call these utilities
*
*****/
{
#ifdef CHECKOUT
printf("\nmlistc ");
#endif

EXEC SQL CLOSE S1;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int mlisto()
/*****
*
* mlisto --- Open cursor S2 for a list of user information
* of navigation points on a particular MFR
*
* Routine executes an open cursor command for cursor S2 and then
*
*****/

```

```

*      returns to the calling program with the ORACLE status code.      *
*      *
*      Note:  Modifications to this function may impact the related    *
*      functions mtlisf() and mtlisr() that fetch rows and             *
*      close the cursor and, possibly, functions that call             *
*      these utility routines.                                          *
*      *
*****/
{
#ifdef CHECKOUT
printf("\mtliso ");
#endif

strcpy( aid.arr, soid.arr);
strcat(aid.arr,"4");
aid.len = strlen(aid.arr);

EXEC SQL DECLARE S2 CURSOR FOR
      SELECT   fix_label, fix_id,      fix_type,      fix_rad,
              fix_dist, width_left, width_right
      FROM     strsegments
      WHERE    fix_label LIKE :aid
      ORDER BY fix_label;

EXEC SQL OPEN S2;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int mtlisf()
/*****
*
*      mtlisf  ---  Fetch a row using the opened cursor S2 for MFR
*      -----  Navigation Point User Information
*
*      Routine executes an fetch command for cursor S2, which is assumed
*      to have been opened, and then returns to the calling program with
*      the ORACLE status code.
*
*      Note:  Modifications to this function may impact the related
*      functions mtliso() and mtlisr() that open and close the
*      cursor and, likely, functions that call these utilities
*
*****/
{
EXEC SQL FETCH S2 INTO :a2bv, :prefixid, :prefixtyp, :prefixrad,
                    :prefixdist, :prewidleft, :prewidright;

prefixid.arr[prefixid.len] = '\0';
prefixtyp.arr[prefixtyp.len] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int mtlisr()
/*****
*
*      mtlisr  ---  Close cursor S2 for Navigation Points
*      -----
*
*      Routine executes a close cursor command for cursor S2 and then
*      returns to the calling program with the ORACLE status code.
*
*      Note:  Modifications to this function may impact the related
*      functions mtliso() and mtlisf() that open the cursor
*      and fetch rows using it and, possibly, functions that
*      call these utilities
*
*****/

```

```

{
#ifdef CHECKOUT
printf("\nm1isto ");
#endif

EXEC SQL CLOSE S2;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int m1isto()
/*****
*
*   m1isto    ---   Open cursor S3 for a list of user information
*   -----          of navigation points on a particular MTR
*
*   Routine executes an open cursor command for cursor S3 and then
*   returns to the calling program with the ORACLE status code.
*
*   Note:   Modifications to this function may impact the related
*           functions m1istf() and m1ista() that fetch rows and
*           close the cursor and, possibly, functions that call
*           these utility routines.
*
*****/
{
#ifdef CHECKOUT
printf("\nm1isto ");
#endif

strcpy( cid.arr, ercid.arr);
strcat(cid.arr, "%");
cid.len = strlen(cid.arr);

EXEC SQL DECLARE S3 CURSOR FOR
    SELECT   fix_label, floor_ref, ceiling_ref, fix_id,   fix_type,
           arcos,   fix_lat,   fix_lon,   fix_rad, fix_dist,
           floor,   ceiling, width_left, width_right
    FROM     mtrsegments
    WHERE    fix_label LIKE :cid
    ORDER BY fix_label;

EXEC SQL OPEN S3;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int m1istf()
/*****
*
*   m1istf    ---   Fetch a row using the opened cursor S3 for MTR
*   -----          Navigation Point User Information
*
*   Routine executes an fetch command for cursor S3, which is assumed
*   to have been opened, and then returns to the calling program with
*   the ORACLE status code.
*
*   Note:   Modifications to this function may impact the related
*           functions m1isto() and m1ista() that open and close the
*           cursor and, likely, functions that call these utilities
*
*****/
{
strcpy(optr3, "          "); /* If you don't use VARCHAR, you have to */
strcpy(optr4, "          "); /* clear the space or weird things happen */

EXEC SQL FETCH S3 INTO :a2bv,      :optr1,      :optr2,      :ourfixid,
                      :ourfixtyp, :ourarctan, :optr3,      :optr4,
                      :ourfixrad, :ourfixdist, :lptr1,      :lptr2,
                      :ourwidthleft, :ourwidthright;

```

```

prefixid.arr[prefixid.len] = '\0';
prefixtyp.arr[prefixtyp.len] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int mlistc()
/*****
 *
 *      mlistc  ---  Close cursor #3 for Navigation Points
 *
 *
 *      Routine executes a close cursor command for cursor #3 and then
 *      returns to the calling program with the ORACLE status code.
 *
 *      Note:  Modifications to this function may impact the related
 *      functions mlistc() and mlistf() that open the cursor
 *      and fetch rows using it and, possibly, functions that
 *      call these utilities
 *
 *****/
{
#ifdef CHECKOUT
printf("\nmlistc ");
#endif

EXEC SQL CLOSE #3;
#ifdef CHECKOUT
printf(" Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

/*****
 *
 *      startup.ps  --  ASAM Initialization Code.  This program is only
 *      used to determine the status of ORACLE and to
 *      load those pieces which are needed.  The space
 *      it occupies can be relinquished after execution.
 *
 *****/

#include <stdio.h>          /* The usual stuff, of course */
#include <process.h>        /* Header for calls to MS-DOS */
#include <string.h>         /* String manipulation header */
#include <time.h>

#define SQLCA_STORAGE_CLASS extern

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE hostvars.h;
EXEC SQL INCLUDE faharris.h;

EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE SQLCA;
#include "asam.h"

startasam()
/*****
 *
 *      startasam  --  start ASAM function
 *
 *
 *      Starts by attempting to connect to ORACLE as "SUPERUSER".
 *      If ORACLE is not up it will attempt to install it.  If
 *      unsuccessful, execution will terminate with appropriate
 *      diagnostic information (i.e. what piece of the ORACLE
 *      database manager it cannot find).
 *
 *
 *      Note:  This program is totally dependent on how ORACLE and its
 *      associated programs behave.  It should be tested (with
 *      #define CHECKOUT here and in other files with ORACLE calls)
 *      when a new ORACLE release is installed.  Particularly the
 *      return codes for uninstalled systems, since they are not
 *
 *****/

```



```

*      according to specification in release 5.1. (e.g., -3120      *
*      should be -3121).                                           *
*                                                                    *
*****/
{
    int rcode, xcode;          /* Oracle return code temporaries */
    int doode = 0;             /* MS-DOS return code */
    int spawlp(), svccan();

    char *clock();

    strcpy(ASSESSMENT.name, "SUPERUSER");

    OO_rcode = svccan();        /* Try SUPERUSER and see what happens */

    if (rcode = OO_rcode) {    /* 0 == ORACLE up and running */

        switch (rcode) {
            case -3120:        /* 3120 == SQLPMS is not installed */
                /* ----- */

                doode = spawlp(P_WAIT, "sqlpms.exe", "sqlpms.exe", "/noior", NULL);
                if ( doode < 0 ) {
                    printf("\nSystem not properly installed! ASAW can't find");
                    printf("\n\nORACLE's Protected Mode Executive SQLPMS.EXE");
                    printf("\nYour Data Administrator should be able to help\n\n");
                    sprintf(dante, "\n\nSQLPMS Failed (%d)", clock(), doode);
                    exit(16);
                }
                else if ( doode < 768 ) {
                    printf("\n\nORACLE's Protected Mode Executive SQLPMS.EXE");
                    printf("\n\nAbnormally terminated. Return code was %d", doode);
                    printf("\nYour Data Administrator should be able to help");
                    sprintf(dante, "\n\nSQLPMS Failed (%d)", clock(), doode);
                    exit(16);
                }
                sprintf(dante, "\n\nSQLPMS Installed", clock());

            case ORA_UNAVAILABLE: /* Not available: Do IOR first */
                /* ----- */
                doode = spawlp(P_WAIT, "ior.exe", "ior.exe", "warm", NULL);
                if ( doode < 0 ) {
                    printf("\nSystem not properly installed! ASAW can't find");
                    printf("\n\nORACLE's Startup routine IOR.EXE");
                    printf("\nYour Data Administrator should be able to help");
                    sprintf(dante, "\n\nORACLE Server Failed (%d)", clock(), doode);
                    exit(16);
                }
                else if ( doode > 0 ) {
                    printf("\n\nORACLE's Initialization Routine IOR.EXE has");
                    printf("\n\nAbnormally terminated. Return code was %d", doode);
                    printf("\nYour Data Administrator should be able to help");
                    sprintf(dante, "\n\nORACLE Server Failed (%d)", clock(), doode);
                    exit(16);
                }
                else {
                    /* Give CONNECT one more try */
                    sprintf(dante, "\n\nORACLE Server Started", clock());
                    if ( xcode = svccan() ) {
                        printf("\n\nASAW has attempted to install the ORACLE Database");
                        printf("\n\nserver, but after what \n\nappears to have been a ");
                        printf("\n\nsuccessful installation ASAW doesn't want to start");
                        printf("\n\nCause: %s", sqlca.sqlwarn.sqlwarnm);
                        if (sqlca.sqlcode == SQL_BAD_LOGIN)
                            printf("\n\nASAW installation program has probably not been run yet");
                        printf("\n\nYour Data Administrator should be able to help\n\n");
                        system("pause");
                        closeORA();
                        sprintf(dante, "\n\nInitial Connect failed second time (%d)",
                                clock(), xcode);
                        exit(16);
                    }
                    break;
                }
            default: {
                sprintf(dante, "\n\nASAW cannot establish communication with ORACLE",
                        clock());
                printf("\n\nCause: %s", sqlca.sqlwarn.sqlwarnm);
                printf("\n\nASAW cannot establish communication with the ORACLE ");
                printf("\n\nDatabase Server. \n\nCause: %s", sqlca.sqlwarn.sqlwarnm);
                if (sqlca.sqlcode == SQL_BAD_LOGIN)
                    printf("\n\nASAW installation program has probably not been run yet");
                printf("\n\nYour Data Administrator should be able to help\n\n");
                system("pause");
            }
        }
    }
}

```

```

        closeORA();
        exit(16);
    } /* End Switch */
} /* End If */
sprintf(dante, "\n%s ASAN Started ", clock());
return 0;
}

```

```

int vfy_ASAN()
/*****
 *
 * vfy_ASAN -- Routine to check ASAN release levels and other system
 * integrity functions. This function is not implemented
 * in nor meaningful for the prototype version of ASAN
 *
 *****/
{
return (int) 0;
}

```

```

int pwcheck()
/*****
 *
 * pwcheck -- Routine to check password and/or user name validity
 * This function is not implemented in the prototype
 *
 *****/
{
return (int) 0;
}

```

```

 *
 * tables.pc -- Set of routines to access the data dictionary's
 *             inventory of users, tables, columns etc.
 *
 * For each ORACLE cursor there are three routines as follows:
 *
 * 1. ...listo() Open Cursor (Opens a "logical file")
 * 2. ...listf() Fetch Cursor (Reads next record from logical file)
 * ...bunch() Fetch Cursor (Reads next bunch from logical file
 * applicable only for multiple choice options)
 * 4. ...listc() Close Cursor (Closes the logical file)
 *
 * Routines in this file:
 *
 * ... = u* - Names of assessments (LIFO order on date started).
 * t - Names and comments of all tables CREATED by current
 * assessment only.
 * c - Names and comments of columns in specific table [:tid]
 * (retrieved in order of column in the table).
 * tc - Names and comments of columns in all tables CREATED by
 * current assessment (sorted alphabetically on table and
 * column within table)
 * v - ASAN table_of_contents in alphabetic order.
 *
 * * - Asterisk indicates that multiple choice option is
 * supported for this set of routines.
 *
 *****/

```

```

#include <stdio.h> /* The usual stuff, of course */
#include <process.h> /* Header for calls to MS-DOS */
#include <string.h> /* String manipulation header */

```

```

#define SQLCA_STORAGE_CLASS extern
EXEC SQL INCLUDE SQLCA; /* SQL Communication Area */

```

```

EXEC SQL BEGIN DECLARE SECTION;

```

```

EXEC SQL INCLUDE hostvars.h;
EXEC SQL INCLUDE %aharris.h;

```

```

EXEC SQL END DECLARE SECTION;

```

```

#include "asan.h" /* Standard ASAN Header file */

```

```

int ulisto()
/*****

```

```

*
*      ulisto  ---  Open cursor U1 for userlist (ASAM assessments)
*      -----
*
*      Routine executes an open cursor command for cursor U1 and then
*      returns to the calling program with the ORACLE status code.
*
*      Note:  Modifications to this function may impact the related
*             functions ulistf(), ubunch() and ulisto() that fetch rows
*             and close the cursor and, possibly, functions that call
*             these utility programs
*
*      *****/
{
#ifdef CHECKOUT
printf("\nulisto ");
#endif

EXEC SQL DECLARE U1 CURSOR FOR
        SELECT username          /* Assessments are ORACLE Users */
        FROM sysuserlist,       /* Standard Data Dictionary view */
             table_of_contents /* ASAM's List of assessments. */
        WHERE sysuserlist.userid = table_of_contents.idnumber
        ORDER BY timestamp DESC; /* LIFO Listing Order */

EXEC SQL OPEN U1;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int ulistf()
/*****
*
*      ulistf  ---  Fetch a row using the opened cursor U1 for userlist
*      -----
*
*      Routine executes an fetch command for cursor U1, which is assumed
*      to have been opened, and then returns to the calling program with
*      the ORACLE status code.
*
*      Note:  Modifications to this function may impact the related
*             functions ulisto(), ubunch() and ulisto() that open,
*             fetch groupwise and close the cursor and, likely, any
*             functions that call these utility programs
*
*      *****/
{
EXEC SQL FETCH U1 INTO :uid;
uid.arr[uid.lam] = '\0';
#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int ubunch()
/*****
*
*      ubunch  ---  Fetch a bunch (20 or whatever the size of default)
*      -----          using the opened cursor U1 for sysuserlist
*
*      Routine executes an fetch command for cursor U1, which is assumed
*      to have been opened, and then returns to the calling program with
*      the ORACLE status code.
*
*      Note:  Modifications to this function may impact the related
*             functions ulisto(), ulistf() and ulisto() that open and
*             close the cursor and, likely, functions that call these
*             utility programs
*
*      *****/
{
register int i;
int rows, rows, reads;
char *clock();

```

```

EXEC SQL FETCH U1 INTO :deplmult;
arows = (int) sqlca.sqlarwd[2];
arows = (sizeof deplmult) / 34;
roode = (int) sqlca.sqlcode;

#ifdef CHECKOUT
printf(" Bunch: %d returns %d of %d rows", sqlca.sqlcode, arows, arows);
#endif
if ((roode == SQL_FETCH_OUT_OF_ORDER) || (roode == SQL_EOF)) {
    if ((roode == SQL_EOF) && (arows > 0)) {
        for (i = 0; i < arows; i++) deplmult[i].arr[deplmult[i].lca] = '\0';
        if (arows < arows)
            for (i = arows; i < arows; i++) deplmult[i].arr[0] = '\0';
        NEWVALS();
        SLOUTP("The last assessment in the list is on the screen");
    }
    else {
        NEWVALS();
        SLOUTP("You are already as far down in the list as you can");
    }
}
else if (roode) sprintf(dante, "\n%s UNRECH: %s",
                        clock(), sqlca.sqlarwd.sqlarwd);
NEWVALS();
return roode;
}

int ulisto()
/*****
 *
 *      ulisto  ---   Close cursor U1 for userlist
 *
 *
 *      Routine executes a close cursor command for cursor U1 and then
 *      returns to the calling program with the ORACLE status code.
 *
 *      Note:  Modifications to this function may impact the related
 *      functions ulisto(), ulistf() and ubunch() that open the
 *      cursor and fetch rows using it and, possibly, functions
 *      that call these utilities
 *
 *****/
{
#ifdef CHECKOUT
printf("\nulisto ");
#endif

EXEC SQL CLOSE U1;
#ifdef CHECKOUT
printf("Close: %d ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int tlisto()
/*****
 *
 *      tlisto  ---   Open cursor U2 for list of tables in an assessment
 *
 *
 *      Routine executes an open cursor command for cursor U2 and then
 *      returns to the calling program with the ORACLE status code.
 *
 *      Note:  Modifications to this function may impact the related
 *      functions tlistf() and tlistc() that fetch rows and
 *      close the cursor respectively and, possibly, functions
 *      that call these utilities
 *
 *****/
{
#ifdef CHECKOUT
printf("\ntlisto");
#endif

EXEC SQL DECLARE U2 CURSOR FOR
        SELECT name, cnt      /* That is what table name and */
        FROM   sys.v4axptab  /* comment are called in this view */
        ORDER BY name;

EXEC SQL OPEN U2;

```

```

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

```

```

int tlistf()
/*****
 *
 * tlistf --- Fetch a row using the opened cursor U2 for tablelist
 *
 *
 * Routine executes an fetch command for cursor U2, which is assumed
 * to have been opened, and then returns to the calling program with
 * the ORACLE status code.
 *
 * Note: Modifications to this function may impact the related
 * functions tlisto() and tlistc() that open and close
 * the cursor respectively and, possibly, functions that
 * call these utilities
 *
 *****/
{
EXEC SQL FETCH U2 INTO :tid, :workspace;
tid.arr[tid.lam] = '\0';
workspace.arr[workspace.lam] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

```

```

int tlistc()
/*****
 *
 * tlistc --- Close cursor U2 for tablelist
 *
 *
 * Routine executes a close cursor command for cursor U2 and then
 * returns to the calling program with the ORACLE status code.
 *
 * Note: Modifications to this function may impact the related
 * functions tlisto() and tlistf() that open the cursor and
 * fetch rows using it respectively and, possibly, functions
 * that call these utilities
 *
 *****/
{
#ifdef CHECKOUT
printf(" tlistc ");
#endif

EXEC SQL CLOSE U2;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

```

```

int clistc()
/*****
 *
 * clistc --- Open cursor U3 for list of columns in a table
 *
 *
 * Routine executes an open cursor command for cursor U3 and then
 * returns to the calling program with the ORACLE status code.
 *
 * Note: Modifications to this function may impact the related
 * functions clistf() and clisto() that fetch rows and
 * close the cursor respectively and, possibly, functions
 * that call these utilities
 *
 *****/
{

```

```

#ifdef CHECKOUT
printf("\nallisto");
#endif

EXEC SQL DECLARE U3 CURSOR FOR
    SELECT  name      /* That is what the column is */
    FROM    col       /* called in this view */
    WHERE   tname = :tid /* For the current table */
    ORDER BY colno;     /* In order of creation */

EXEC SQL OPEN U3;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int alistf()
/*****
 *
 *      alistf  ---  Fetch a row using the opened cursor U3 for column list
 *
 *
 *      Routine executes an fetch command for cursor U3, which is assumed
 *      to have been opened, and then returns to the calling program with
 *      the ORACLE status code.
 *
 *      Note:  Modifications to this function may impact the related
 *      functions alist() and alistf() that open and close
 *      the cursor respectively and, possibly, functions that
 *      call these utilities
 *
 *****/
{
EXEC SQL FETCH U3 INTO :aid;
tid.arr[tid.les] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int alistc()
/*****
 *
 *      alistc  ---  Close cursor U3 for tabllist
 *
 *
 *      Routine executes a close cursor command for cursor U3 and then
 *      returns to the calling program with the ORACLE status code.
 *
 *      Note:  Modifications to this function may impact the related
 *      functions alist() and alistf() that open the cursor and
 *      fetch rows using it respectively and, possibly, functions
 *      that call these utilities
 *
 *****/
{
#ifdef CHECKOUT
printf(" alistc ");
#endif

EXEC SQL CLOSE U3;

#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int talisto()
/*****
 *
 *      talisto  ---  Open cursor U4 for list of all columns sorted on table
 *
 *****/

```

```

* ----- *
* Routine executes an open cursor command for cursor U4 and then *
* returns to the calling program with the ORACLE status code. *
* *
* Note: Modifications to this function may impact the related *
* functions talistf() and talista() that fetch rows and *
* close the cursor respectively and, possibly, functions *
* that call these utilities *
* *
*****/

{
#ifdef CHECKOUT
printf("\ntalisto");
#endif

EXEC SQL DECLARE U4 CURSOR FOR
        SELECT tname,      /* TNAME is the table name */
               cname,      /* CNAME that of the column */
               remarks      /* in the COL view of the */
        FROM col           /* ORACLE Data Dictionary */
        ORDER BY tname, cname;

EXEC SQL OPEN U4;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int talistf()
/*****
*
* talistf --- Fetch a row using the opened cursor U4
*          (column names sorted by table name)
*
* Routine executes an fetch command for cursor U4, which is assumed
* to have been opened, and then returns to the calling program with
* the ORACLE status code.
*
* Note: Modifications to this function may impact the related
* functions talisto() and talista() that open and close
* the cursor respectively and, possibly, functions that
* call these utilities
*
*****/
{
EXEC SQL FETCH U4 INTO :tid, :cid, :workspace;
tid.arr[tid.lam] = '\0';
cid.arr[cid.lam] = '\0';
workspace.arr[workspace.lam] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int talista()
/*****
*
* talista --- Close cursor U4 for column list sorted by table
*
* Routine executes a close cursor command for cursor U4 and then
* returns to the calling program with the ORACLE status code.
*
* Note: Modifications to this function may impact the related
* functions talisto() and talistf() that open the cursor and
* fetch rows using it respectively and, possibly, functions
* that call these utilities
*
*****/
{
#ifdef CHECKOUT

```

```

printf(" telists ");
#endif

EXEC SQL CLOSE U4;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int vlisto()
/*****
 *
 *      vlisto  ---  Open cursor U5 for list of ASAN's assessments
 *
 *
 *      Routine executes an open cursor command for cursor U5 and then
 *      returns to the calling program with the ORACLE status code.
 *
 *      Note:  Modifications to this function may impact the related
 *             functions vlistf() and vlists() that fetch rows and close
 *             the cursor and, possibly, functions that call these
 *             utility programs
 *
 *****/
{
#ifdef CHECKOUT
printf("\nvlisto ");
#endif

EXEC SQL DECLARE U5 CURSOR FOR
        SELECT username, description
        FROM sysuserlist,
        table_of_contents
        WHERE sysuserlist.userid = table_of_contents.idnumber
        ORDER BY username DESC;

EXEC SQL OPEN U5;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int vlistf()
/*****
 *
 *      vlistf  ---  Fetch a row using the opened cursor U5 for list of
 *                  ASAN assessments and their descriptions
 *
 *
 *      Routine executes an fetch command for cursor U5, which is assumed
 *      to have been opened, and then returns to the calling program with
 *      the ORACLE status code.
 *
 *      Note:  Modifications to this function may impact the related
 *             functions vlisto() vlists() that open close the cursor
 *             and, likely, any functions that call these utility programs
 *
 *****/
{
EXEC SQL FETCH U5 INTO :uid, :workspace;
uid.arr[uid.len] = '\0';
workspace.arr[workspace.len] = '\0';
#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int vlists()
/*****
 *
 *      vlists  ---  Close cursor U5 for ASAN table_of_contents
 *
 *****/

```



```

*
* Routine executes a close cursor command for cursor US and then
* returns to the calling program with the ORACLE status code.
*
*
* Note: Modifications to this function may impact the related
*       functions ulisto() and ulistf() open the cursor and fetch
*       rows using it and, possibly, functions that call these
*       utilities
*
*
*****/
{
#ifdef CHECKOUT
printf("\nvlsto ");
#endif

EXEC SQL CLOSE US;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

/*****
*
* util.pc -- A few general utilities
*
*
* This file contains:
*
* expMsg    - displays expansion of ORACLE error message
* expWarn   - displays expansion of ORACLE warning message
* clock     - returns time/date information
*
*****/

#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */
EXEC SQL BEGIN DECLARE SECTION; /* All SQL declarations for this */
EXEC SQL INCLUDE hostvars.h;    /* are in these two header files */
EXEC SQL INCLUDE sbharris.h;    /* this one comes from "U" */
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE sqlca;
#include "asaa.h"             /* Standard ASAA Header File */

void expMsg()
/*****
*
* expMsg -- Expand Oracle Error Message
*
*****/
{
#ifdef CHECKOUT
printf("\t%.70s\n", sqlca.sqlerrm.sqlerrmc);
#else
EXEC MSG;
SLOCUS(sqlca.sqlerrm.sqlerrmc);
#endif
}

void expWarn()
/*****
*
* expWarn -- Expand Oracle Warning Message
*
*****/
{
if (sqlca.sqlwarn[1] == 'W')
SLOCUS("SQLWARNING: Column Truncated");
if (sqlca.sqlwarn[2] == 'W')
SLOCUS("SQLWARNING: Null in aggregate");
if (sqlca.sqlwarn[3] == 'W')
SLOCUS("SQLWARNING: INFO var count != col count");
}

```

```

if (sqlca.sqlwarn[4] == 'W')
    SLOCUTP("SQLWARNING: Update or Delete w/o WHERE");
if (sqlca.sqlwarn[5] == 'W')
    SLOCUTP("SQLWARNING: ?????");
if (sqlca.sqlwarn[6] == 'W')
    SLOCUTP("SQLWARNING: Rollback Required");
if (sqlca.sqlwarn[7] == 'W')
    SLOCUTP("SQLWARNING: Change after query for UPDATE");
}

char *clock()
/*****
 *
 *      clock  -- Routine that returns the location of the string
 *      ===== with the date and time of this very moment
 *
 *****/
{
    long now, time();
    char *ctime(), *text;

#ifdef CHECKOUT
    printf("\nclock ");
#endif

    now = time(NULL);
    text = ctime(&now);
    text[24] = ' ';
    return text;
}

VCAPITAL(x)
/*****
 *
 *      VCAPITAL  -- Convert a VARCHAR entered by user to uppercase
 *      ===== and store len
 *
 *****/
VARCHAR *x;
{
    register int i, j;
    j = x->len = strlen(x->arr);
    for (i=0; i<j; i++) x->arr[i] = toupper(x->arr[i]);
}

/*****
 *
 *      These are a set of dummy routines while the program is under
 *      development. They will be removed in the production version.
 *
 *****/

dummy()
{
    SLOCUTP("This feature is not available in this prototype version");
}

dummy2()
{
    SLOCUTP("This feature currently runs as a separate program.");
}
/*****
 *
 *      utmacov.pc  -- Routines dealing with the wonderful world of UTM,
 *      =====   DLS, GRASS and such
 *
 *
 *      This file contains:
 *
 *      1. utm to lat/lon conversion (OC_u2ll) routines
 *      2. lat/lon to utm conversion (OC_ll2u) routines
 *
 *      OC_u2ll_spheroid (spheroid_name) must be called first. sets the
 *      spheroid parameters for the ellipse 'spheroid_name' (see

```

```

*      get_spheroid.c for known spheroids).  Function is used for
*      conversions in either direction.
*
*      CC_u2ll_spheroid_parameters (a,e) called by CC_u2ll_spheroid() to
*      set the ellipsoid major axis 'a' and eccentricity squared 'e'
*      can be called directly for unknown ellipsoids.
*
*      CC_u2ll_zone (zone) must be called before CC_u2ll_north(). Set
*      utm 'zone' (must be non-zero). Negative means southern
*      hemisphere. Used to set the longitude of the central
*      meridian (only used for utm to lat/lon conversions)
*
*      CC_u2ll_north (north) set the utm north. Must be called before
*      CC_u2ll.
*
*      CC_u2ll (east, lat, lon) computes lat, lon from east after
*      CC_u2ll_north() has already been called with north.
*
*
*      3. miscellaneous user interface service routines
*
*      lon2dec - Convert longitude of COORDINATE variable to seconds
*               decimal
*      lat2dec - Convert latitude of COORDINATE variable to seconds
*               decimal
*      alt2dec - Convert an ALT/SPEC VARIABLE's spec to units and altitude
*      convcoord - Convert the spherical coordinate character string of
*                 a COORDINATE variable to decimal
*      COOR2utm - Convert COORDINATE decimal lat/lon to utm
*
*      *****/

#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>
#include <ctype.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION; /* All SQL declarations are in */
EXEC SQL INCLUDE hostvars.h;    /* these header files */
EXEC SQL INCLUDE sakaris.h;
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asan.h"              /* Standard ANSI Header File */

#define abs(x) ((x)<0?-(x):(x))

#define RADIANS_TO_SECONDS 206264.8062470964
#define SECONDS_TO_RADIANS 4.84813681109536e-6

double sqrt(), sin(), cos();

static double a1,a2,a3,a4; /* lat coef: geodetic to rectifying */
static double a5 = 5.0e5; /* false easting (UTM) */
static double a6; /* false northing */
static double a7 = 0; /* MO? */
static double a8 = .9996; /* UTM scale factor at central meridian */
static double a9; /* central meridian in seconds */
static double a10; /* radius of curvature */
static double a11,a12,a13,a14; /* lat coef: rectifying to geodetic */
static double a15; /* major axis */
static double a16; /* eccentricity squared */

static double b1,b2,b3,b4; /* intermediate values */
static double b5,b6,b7,b8;
static double b9,b10,b11,b12;

static struct { char *name;
                double a; /* semi-major axis */
                double e; /* eccentricity squared */
                } spheroid[] =
{
    "australian", 6378160.0, 0.0066945419,
    "bessel",     6377739.155, 0.0066743722,
    "clark66",    6378206.4, 0.006746658,
    "clark80",    6378249.145, 0.0068035113,
    "everest",    6377276.345, 0.0066378466,
    "international", 6378388.0, 0.00672267,
    "wgs72",     6378135.0, 0.006694317778
};

```

```

int OC_get_spheroid (name, a, e)
/*****
 *
 *   OC_get_spheroid  --  Returns axis and eccentricity squared
 *                       for the named spheroid. Return value
 *                       1 for success, 0 for failure to find
 *                       the spheroid in the list
 *
 *****/

```

```

char *name;
double *a, *e;
{
int n;

n = sizeof(spheroid)/sizeof(spheroid[0]);

#ifdef CHECKROOT
printf("\nget_spheroid %s from %d entries ", name, n);
#endif

```

```

while (n--)
if (equal (name, spheroid[n].name)) {
*a = spheroid[n].a;
*e = spheroid[n].e;
return 1;
#ifdef CHECKROOT
printf(" found %d", n);
#endif
}

#ifdef CHECKROOT
printf(" not found");
#endif
return 0 ;
}

```

```

char * OC_spheroid_name (n)
/*****
 *
 *   OC_spheroid_name  --  Find the name of spheroid n in the list
 *
 *****/

```

```

int n;

{
if (n < 0 || n >= sizeof(spheroid)/sizeof(spheroid[0])) return 0;
else return spheroid[n].name;
}

```

```

static equal (a, b)
/*****
 *
 *   equal  --  comparison between two character strings
 *             (Forced to lowercase)
 *
 *****/
char *a, *b;
{
char lcase();

while (*a) if (lcase (*a++) != lcase (*b++)) return 0;
return *b == 0;
}

```

```

static char lcase (c)
/*****
 *
 *   lcase  --  Force character to lowercase
 *
 *****/

```

```

*****/
char c;
{
if (c >= 'A' && c <= 'Z') c += 'a' - 'A';
return c;
}

int OC_u21l_spheroid (spheroid_name)
/*****
*
*   OC_u21l_spheroid -- Set up spheroid parameters for UTM to Lat/Long
*   given a spheroid name
*
*****/
char *spheroid_name ;
{
double a,e;

if (OC_get_spheroid (spheroid_name, &a, &e)) /* Known Spheroid */
return OC_u21l_spheroid_parameters (a, e);
return -1; /* Unknown Spheroid */
}

int OC_u21l_spheroid_parameters (a,e)
/*****
*
*   OC_u21l_spheroid_parameters -- Store the conversion parameters
*   given axis and eccentricity
*
*****/
double a,e;
{
double x,x2,x3,x4;

if (a < 0.0 || e < 0.0 || e > 1.0) return -2; /* illegal values */

a15 = a;
a16 = e;

x = (((a * (7.0/32.0) + 5.0/16.0) * e + .5) * e + 1.0) * e * .25;
x2 = x * x;
x3 = x * x2;
x4 = x * x3;

/* coefficients to convert geodetic to rectifying latitude */
a1 = -(((x * (195.0/64.0) + 3.25) * x + 3.75) * x + 3.0) * x;
a2 = (((1455.0/32.0) * x + 70.0/3.0) * x + 7.5) * x2;
a3 = -((70.0/3.0) + x * (945.0/8.0)) * x3;
a4 = (315.0/4.0) * x4;

/* coefficients to convert rectifying to geodetic latitude */
a11 = (((7.75 - (657.0/64.0) * x) * x - 5.25) * x + 3.0) * x;
a12 = (((5045.0/32.0) * x - (151.0/3.0)) * x + 10.5) * x2;
a13 = ((151.0/3.0) - (3291.0/8.0) * x) * x3;
a14 = (109.0/4.0) * x4;

/* radius of curvature */
a10 = (((225.0/64.0) * x2 + 2.25) * x2 + 1.0) * (1.0 - x2) * (1.0 - x) * e;
return 1;
}

int OC_u21l_zone (zone)
/*****
*
*   OC_u21l_zone
*
*****/

```

```

* ----- *
*
* *****/
{
double uts;

/* set false northing (a6), compute central meridian (a9) */

if (zone < 0) {
a6 = 10.0e6;
uts = 30.0 + zone;
}
else {
a6 = 0.0;
uts = 30.0 - zone;
}
a9 = (uts * 5.0 + 3.0) * 3600.0;
}

int CC_u211_north (north)
/*****
*
* CC_u211_north
*
* *****/
double north;
{
double sinw, cosw;
double t, ts, ra, ra2, ra4, ra6, ra8;
double etas;

b10 = ((north - a6) / a8 + a7) / a10;
if (abs(b10) > 1.47)
return -1; /* rectifying lat exceeds 1.47 radians, -84.15.30 */

sinw = sin(b10);
cosw = cos(b10);
b12 = cosw * cosw;
b11 = (((a14 * b12 + a13) * b12 + a12) * b12 + a11) * sinw * cosw + b10;

sinw = sin(b11);
cosw = cos(b11);
ra = sqrt (1.0 - a16*sinw*sinw) * 1.0e6 / a15;
ra2 = ra * ra;
ra4 = ra2 * ra2;
ra6 = ra2 * ra4;
ra8 = ra4 * ra4;
t = sinw/cosw;
ts = t * t;

b12 = cosw * cosw;
etas = a16 * b12 / (1.0 - a16);

b1 = ra/cosw;
b2 = -t * (1.0 + etas) * ra2 / 2.0;
b3 = - (1.0 + ts + ts + etas) * b1 * ra2 / 6.0;
b4 = (((-6.0 - etas * 9.0) * etas + 3.0) * ts +
(6.0 - etas * 3.0) * etas + 5.0) * t * ra4 / 24.0;
b5 = ((ts * 24.0 + etas * 6.0 + 28.0) * ts
+ etas * 6.0 + 5.0) * b1 * ra4 / 120.0;
b6 = (((etas * 45.0 - 45.0) * ts + etas * 162.0 - 90.0) * ts
- etas * 107.0 - 61.0) * t * ra6 / 720.0;
b7 = -(((ts * 720.0 + 1320.0) * ts
+ 662.0) * ts + 61.0) * b1 * ra6 / 5040.0;
b8 = (((ts * 1575.0 + 4095.0) * ts
+ 3633.0) * ts + 1385.0) * t * ra8 / 40320.0;

return 1;
}

int CC_u211 (east, lat, lon)
/*****
*
* CC_u211
*
* *****/

```

```

*
*****/

double east;
double *lat, *lon;
{
    double b9, b10;

    b9 = ((a5 - east) * 1.0e-6) / a8 ;
    if (abs(b9) > a15 * 2.0e-7)
        return -1; /* uia easting to far from center of zone */

    b10 = b9 * b9 ;
    *lat = (((b9 * b10 + b6) * b10 + b4) * b10 + b2) * b10 + b11)
        * RADIANS_TO_SECONDS ;
    *lon = (((b7 * b10 + b5) * b10 + b3) * b10 + b1) * b9
        * RADIANS_TO_SECONDS + a9 ;

    return 1;
}

int OC_112u (lat, lon, east, north, zone)
/*****
*
*   OC_112u
*
*
*****/

double lat, lon;
double *east, *north;
int *zone;
{
    int day;
    double simp, coesp;
    double etas, t, ts, ra;
    double c1, c2, c3;

    /* if zone is != 0, force into this zone, otherwise compute the zone */
    if (*zone == 0) {
        if (lon < 0) { /* eastern zones */
            day = -lon / 3600;
            *zone = 31 + day / 6;
        }
        else { /* western zones */
            day = lon / 3600;
            *zone = 30 - day / 6;
        }
        if (lat < 0) *zone = -(*zone);
    }

    /* now, set a6, a9 */

    OC_u11l_zone (*zone);
    if (abs(lat) > 302400.0) return -1; /* latitude above 84 degrees */

    b10 = (a9 - lon) * SECONDS_TO_RADIANS;
    if (abs(b10) > .16) return -2; /* longitude to far from center of wta zone */

    b9 = lat * SECONDS_TO_RADIANS;
    simp = sin (b9);
    coesp = cos (b9);
    ra = a15 / sqrt (1.0 - a16 * simp * simp);
    t = simp / coesp;
    ts = t * t;
    c1 = coesp * coesp;
    c2 = c1 * c1;
    c3 = c1 * c2;
    etas = a16 * c1 / (1.0 - a16);

    b1 = ra * coesp;
    b3 = (1.0 - ts + etas) * b1 * c1 / 6.0;
    b5 = ((ts - 16.0) * ts + 5.0 + (14.0 - 59.0 * ts) * etas) * b1 * c2 / 120.0;
    b7 = (((179.0 - ts) * ts - 479.0) * ts + 61.0) * b1 * c3 / 5040.0;
    b12 = b10 * b10;

    *east = (((b7 * b12 + b5) * b12 + b3) * b12 + b1) * b10 * a8 + a5;

```

```

b2 = xm * c1 * t / 2.0;
b4 = (etas * (9.0 + 4.0 * etas) + 5.0 - ts) * b2 * c1 / 12.0;
b6 = ((ts - 59.0) * ts + 61.0 +
      (270.0 - 333.0 * ts) * etas) * b2 * c2 / 360.0;
b8 = ((543.0 - ts) * ts - 3111.0) * ts + 1385.0) * b2 * c3 / 20160.0;

*north = (((b8 * b12 + b6) * b12 + b4) * b12 + b2) * b12 +
          (((a4 * a1 + a3) * a1 + a2) * a1 + a1) * simp * coep + b9)
          * a10;

*north = (*north - a7) * a8 + a6;

return 1;
}

int lon2dec(x)
/*****
*
*      lon2dec  -- Convert a COORDINATE VARIABLE's longitude to
*      decimal (seconds of arc)
*
*      Routine unpacks longitude of the structure, reformats it and then
*      stores the reformatted and the converted value in the structure
*
*****/
COORDINATE *x;

{
double coavcoord();
register int i;

#ifdef CHECKOUT
printf("\nlon2dec ");
#endif

for (i = 0; i < 14; i++) x->lon[i] = toupper(x->lon[i]);
if ((x->longitude = coavcoord(x->lon)) < OFF_EARTH) {
    if (x->longitude > 0.0) x->lon[12] = 'W';
    else x->lon[12] = 'E';
}
else x->eastings = -1.0e75;
}

int lat2dec(x)
/*****
*
*      lat2dec  -- Convert a COORDINATE VARIABLE's latitude to
*      decimal (seconds of arc)
*
*      Routine unpacks latitude of the structure, reformats it and then
*      stores the reformatted and the converted value in the structure
*
*****/
COORDINATE *x;

{
double coavcoord();
register int i;

for (i = 0; i < 14; i++) x->lat[i] = toupper(x->lat[i]);
x->latitude = coavcoord(x->lat);
if (x->latitude < OFF_EARTH) {
    x->lat[0] = ' ';
    if (x->latitude > 0.0) {
        if (x->latitude > 324000.0) {
            SLOUTP(" > 90 ");
            x->northings = -1.0e75;
        }
        else x->lat[12] = 'N';
    }
    else {
        if (x->latitude < -324000.0) {
            SLOUTP(" > 90 ");
            x->northings = -1.0e75;
        }
        else x->lat[12] = 'S';
    }
}
}

```



```

    }
    else x->northings = -1.0e75;
}

```

```

int alt2dec(x)
/*****
 *
 *      alt2dec  -- Convert an ALTSPAC VARIABLE's spec to units and
 *      _____ long integer value
 *
 *      Note: Use only to convert values at the time of entry or the status
 *      line message will make you wonder with great amazement .....
 *
 *****/
ALTSPAC *x;
{
    register int i, j;

    for (i = 0; i < 10; i++) x->spec[i] = toupper(x->spec[i]);

    j = strlen(x->spec);
    x->altitude = 0;

    for (i = 0; i < j; i++) {
        if (isspace(x->spec[i])) continue;
        if (isdigit(x->spec[i])) break;
        if (x->spec[i] == 'A') {
            strcpy(x->spec, "AS ASSG'D ");
            strcpy(x->units, "TMD");
        }
        else SLOUTP("Not altitude");
        return 0;
    }
    if (i == j) {
        SLOUTP("No value");
        return 0;
    }
    do {
        x->altitude = (10 * x->altitude) + x->spec[i] - 48;
        i++; } while (isdigit(x->spec[i]) && i < j);
    for (; i < j; i++)
        if (!isspace(x->spec[i])) break;
    if (i == j) {
        SLOUTP("Need Reference (MSL, AGL, SFC) or ASSIGNED");
        return 0;
    }
    if (x->spec[i] == 'A') strcpy(x->units, "AGL");
    else if (x->spec[i] == 'M') strcpy(x->units, "MSL");
    else if (x->spec[i] == 'S') strcpy(x->units, "SFC");
    else return 0;
    sprintf(x->spec, "%04ld %s", x->altitude, x->units);
    return 0;
}

```

```

double convcoord(x)
/*****
 *
 *      convcoord  -- Unpack a character string with a spherical
 *      _____ coordinate and return seconds of arc as a double
 *
 *      Note: The character string will be reformatted to xxxxx'aa.b"X
 *      (as in 112x30'24.5"W)
 *
 *****/

char x[];
{
    int i, j, ideg = 0, imin = 0;
    double t = 0.0;

    j = strlen(x);
    for (i = 0; i < j; i++) {
        if (isspace(x[i])) continue;
        if (isdigit(x[i])) break;
        SLOUTP("Not a oordinate");
        return (double) OFF_EARTH;
    }
    if (i == j) {
        SLOUTP("No value");
        return (double) OFF_EARTH;
    }
}

```

```

do {
    iddeg = (10 * iddeg) + x[i] - 48;
    i++;} while (isdigit(x[i]) && i < j);

if ((iddeg > 180) ||
    ((x[i] != ' ') && (x[i] != 'D') && (x[i] != '\0') && (x[i] != -8))) {
    SLOUTP("? Degrees"); return (double) OFF_EARTH;}

if (i < j-1) {
    i++;
    for (; i < j; i++) {                /* Minutes */
        if (isspace(x[i])) continue;
        if (isdigit(x[i])) break;
        SLOUTP("? Minutes");
        return (double) OFF_EARTH;}
    for (; isdigit(x[i]) && i < j; i++) {
        imin = (10 * imin) + x[i] - 48;
    }

    if ((imin > 60) ||
        ((x[i] != ' ') && (x[i] != 'M') && (x[i] != '\0') && (x[i] != -8))) {
        SLOUTP("? Minutes"); return (double) OFF_EARTH;}
    }

if (i < j-1) {
    i++;
    for (; i < j; i++) {                /* Seconds */
        if (isspace(x[i])) continue;
        if (isdigit(x[i])) break;
        SLOUTP("? Seconds");
        return (double) OFF_EARTH;}
    for (; isdigit(x[i]) && i < j; i++) {
        t = 10.0 * t + (double) (x[i]-48);
    }
    if (x[i] == '.') {
        i++;
        if (i < j) {
            if (isdigit(x[i])) t += ((double) (x[i]-48))/10.0;
            i++;
        }

        if ((t > 60.0) ||
            ((x[i] != '\0') && (x[i] != ' ') && (x[i] != 'S') && (x[i] != -8))) {
                SLOUTP("? Seconds"); return (double) OFF_EARTH;}
        }

if (i < j) i++;
for (; i < j; i++) {                /* Hemisphere */
    if (isspace(x[i])) break;}

if ((i == j) || (x[i] == 'N') || (x[i] == 'W')) {
    sprintf(x, "%03d %02d' %04.1lf\0", iddeg, imin, t);
    return (3600.0 * (double) (iddeg) + 60.0 * (double) (imin) + t);}
else if ((x[i] == 'S') || (x[i] == 'E')) {
    sprintf(x, "%03d %02d' %04.1lf\0", iddeg, imin, t);
    return (-(3600.0 * (double) (iddeg) + 60.0 * (double) (imin) + t));}
SLOUTP("? Hemisphere");

return (double) OFF_EARTH;
}

```

```

int COOR2utm(x)
/*****
 *
 *      COOR2utm -- Convert a decimal latitude and longitude to the
 *      ----- corresponding utm northing/eastings for an ASAM
 *      COORDINATE structure
 *
 *****/

COORDINATE *x;

{
    char *clock();

if ( (x->latitude < OFF_EARTH) && (x->longitude < OFF_EARTH) ) {
    if (CC_112a (x->latitude, x->longitude,
        &x->eastings, &x->northings, &x->zone) == 1) return (int) 0;

```



```

printf("ASAM Citation module starting.....");
MSA_DEBUG_FEATURES = 1;

strcpy(cituid.arr, "INQUIRYER/ALIGNMENT");
cituid.len = strlen(cituid.arr);

EXEC SQL CONNECT :cituid;

if (sqlca.sqlcode) {
    MESSAGES:
    printf("\nCannot Start Citation Database Inquiry due to\n%s\n",
        sqlca.sqlerrm.sqlerrmc);
    exit (16);
}

EXEC SQL SELECT COUNT(entry_num)
    FROM headquarters.animal_list
    INTO :maxanimals;

Visit();
EXEC SQL ROLLBACK WORK RELEASE;
exit(0);
}
/*****
 *
 * query1.pc -- Routines for citation database retrieval without keywords
 *
 * This file contains:
 *
 * qsetup - Set up for a new search
 * queryh - Select citations from the Human area
 * querya - Select citations from the Animal area
 * querys - Select citations from the Structures area
 * querym - Select citations from the Modeling area
 * queryw - Select based on (first few letters of) Writer's name
 * queryd - Select based on the range of Dates given
 * queryt - Select based on the occurrence of a phrase in the Title
 *
 *****/

#include <process.h> /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asam.h" /* Standard ASAM Header File */

EXEC SQL DECLARE C100 CURSOR FOR
    SELECT entry_num
    FROM headquarters.citation_search
    WHERE human_area = 'T';

EXEC SQL DECLARE C101 CURSOR FOR
    SELECT q.entry_num
    FROM headquarters.citation_search s, qual_citi q
    WHERE s.entry_num = q.entry_num
    AND s.human_area = 'T';

EXEC SQL DECLARE C102 CURSOR FOR
    SELECT q.entry_num
    FROM headquarters.citation_search s, qual_cit2 q
    WHERE s.entry_num = q.entry_num
    AND s.human_area = 'T';

EXEC SQL DECLARE C200 CURSOR FOR
    SELECT entry_num
    FROM headquarters.citation_search
    WHERE animal_area = 'T';

EXEC SQL DECLARE C201 CURSOR FOR

```

```

SELECT q.entry_num
FROM headquarters.citation_search s, qual_cit1 q
WHERE s.entry_num = q.entry_num
AND s.anal_area = 'T';

EXEC SQL DECLARE C202 CURSOR FOR
SELECT q.entry_num
FROM headquarters.citation_search s, qual_cit2 q
WHERE s.entry_num = q.entry_num
AND s.anal_area = 'T';

EXEC SQL DECLARE C300 CURSOR FOR
SELECT entry_num
FROM headquarters.citation_search
WHERE struc_area = 'T';

EXEC SQL DECLARE C301 CURSOR FOR
SELECT q.entry_num
FROM headquarters.citation_search s, qual_cit1 q
WHERE s.entry_num = q.entry_num
AND s.struc_area = 'T';

EXEC SQL DECLARE C302 CURSOR FOR
SELECT q.entry_num
FROM headquarters.citation_search s, qual_cit2 q
WHERE s.entry_num = q.entry_num
AND s.struc_area = 'T';

EXEC SQL DECLARE C400 CURSOR FOR
SELECT entry_num
FROM headquarters.citation_search
WHERE model_area = 'T';

EXEC SQL DECLARE C401 CURSOR FOR
SELECT q.entry_num
FROM headquarters.citation_search s, qual_cit1 q
WHERE s.entry_num = q.entry_num
AND s.model_area = 'T';

EXEC SQL DECLARE C402 CURSOR FOR
SELECT q.entry_num
FROM headquarters.citation_search s, qual_cit2 q
WHERE s.entry_num = q.entry_num
AND s.model_area = 'T';

EXEC SQL DECLARE C500 CURSOR FOR
SELECT authorum
FROM headquarters.author_list a
WHERE UPPER(author) LIKE :pattern;

EXEC SQL DECLARE C500 CURSOR FOR
SELECT entry_num
FROM headquarters.author_citation_link
WHERE authorum = :authorum;

EXEC SQL DECLARE C501 CURSOR FOR
SELECT q.entry_num
FROM headquarters.author_citation_link l, qual_cit1 q
WHERE l.entry_num = q.entry_num
AND l.authorum = :authorum;

EXEC SQL DECLARE C502 CURSOR FOR
SELECT q.entry_num
FROM headquarters.author_citation_link l, qual_cit2 q
WHERE l.entry_num = q.entry_num
AND l.authorum = :authorum;

EXEC SQL DECLARE C600 CURSOR FOR
SELECT entry_num
FROM headquarters.citation_search

```

```

WHERE date_pub >= :date1
AND date_pub <= :date2;

EXEC SQL DECLARE C601 CURSOR FOR
SELECT q.entry_num
FROM headquarters.citation_search s, qual_cit1 q
WHERE s.entry_num = q.entry_num
AND s.date_pub >= :date1
AND s.date_pub <= :date2;

EXEC SQL DECLARE C602 CURSOR FOR
SELECT q.entry_num
FROM headquarters.citation_search s, qual_cit2 q
WHERE s.entry_num = q.entry_num
AND s.date_pub >= :date1
AND s.date_pub <= :date2;

EXEC SQL DECLARE C700 CURSOR FOR
SELECT entry_num
FROM headquarters.citation_titles
WHERE UPPER(title) LIKE :pattern;

EXEC SQL DECLARE C701 CURSOR FOR
SELECT q.entry_num
FROM headquarters.citation_titles s, qual_cit1 q
WHERE s.entry_num = q.entry_num
AND UPPER(s.title) LIKE :pattern;

EXEC SQL DECLARE C702 CURSOR FOR
SELECT q.entry_num
FROM headquarters.citation_titles s, qual_cit2 q
WHERE s.entry_num = q.entry_num
AND UPPER(s.title) LIKE :pattern;

int qsetup()
/*****
*
*      qsetup  -- start-up routine that clears all temporary tables
*      -----  for intermediate pointers in queries
*
*****/
{
#ifdef CHECKOUT
printf("\nqsetup ");
#endif

SLOUT("Purging obsolete qualifier lists");

EXEC SQL DELETE FROM QUAL_CIT1;
if (sqlca.sqlcode)
if (sqlca.sqlcode != SQL_EOF) {
MESSAGE;
SLOUT("%s", sqlca.sqlwarn.sqlwarn);
exit(16);}
#ifdef CHECKOUT
else printf("\nqualifying table1 cleared %ld", sqlca.sqlcode);
#endif

EXEC SQL DELETE FROM QUAL_CIT2;
if (sqlca.sqlcode)
if (sqlca.sqlcode != SQL_EOF) {
MESSAGE;
SLOUT("%s", sqlca.sqlwarn.sqlwarn);
exit(16);}
#ifdef CHECKOUT
else printf("\nqualifying table2 cleared %ld", sqlca.sqlcode);
#endif
#ifdef
EXEC SQL COMMIT WORK;
qual_entries = 0; /* No qualifying entries */
temptabl = 0; /* No temporary table */
dates1 = dates2 = 0; /* No date */
salcrit[0] = '\0'; /* No search path */
animals = mammals; /* ALL Animals ... */
olddpth = 0;
ccntasm[0] = '\0';

```

```

majornt[0] = '\0';
minornt[0] = '\0';
address[0] = '\0';
setaff(4);

```

```

HWVALS();
return (int) 0;
}

```

```

int queryh()
/*****
 *
 *   queryh -- Create the subset of citations for the titles that
 *   -----   pertain to the HUMAN area
 *
 *****/
{
    int i;

#ifdef CHECKOUT
    printf("\nqueryh ");
#endif

    SLOUT("Human Area Search");
    stream(scalrit, "Human ");
    switch (temptabl) {
        case 0:
            EXEC SQL OPEN C100;
            if (sqlca.sqlcode) {
                ENDMSG;
                SLOUT("CURSOR C100");
                SLOUT("%s", sqlca.sqlwarn.sqlwarn);
            }
            qual_entries = 0;
            for (;;) {
                EXEC SQL FETCH C100 into :caumb;
                if (sqlca.sqlcode == SQL_EOF) break;
                if (sqlca.sqlcode) {
                    ENDMSG;
                    SLOUT("CURSOR C100");
                    SLOUT("%s", sqlca.sqlwarn.sqlwarn);
                }
                EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:caumb);
                if (sqlca.sqlcode) {
                    ENDMSG;
                    SLOUT("%s", sqlca.sqlwarn.sqlwarn);
                }
#ifdef CHECKOUT
                printf("\nAccepted ");
                for (i = 0; i < 5; i++) printf("%c", caumb.arr[i]);
#endif
                qual_entries++;
                HWVALS();
            }
            EXEC SQL COMMIT WORK;
            EXEC SQL CLOSE C100;
            temptabl = 1;
            break;

        case 1:
            EXEC SQL DELETE FROM QUAL_CIT2;
            EXEC SQL COMMIT WORK;
            EXEC SQL OPEN C101;
            if (sqlca.sqlcode) {
                ENDMSG;
                SLOUT("CURSOR C101");
                SLOUT("%s", sqlca.sqlwarn.sqlwarn);
            }
            qual_entries = 0;
            for (;;) {
                EXEC SQL FETCH C101 into :caumb;
                if (sqlca.sqlcode == SQL_EOF) break;
                if (sqlca.sqlcode) {
                    ENDMSG;
                    SLOUT("CURSOR C101");
                    SLOUT("%s", sqlca.sqlwarn.sqlwarn);
                    exit(16);
                }
                EXEC SQL INSERT INTO QUAL_CIT2 (entry_num) VALUES (:caumb);
            }

```

```

        if (sqlca.sqlcode) {
            MESSAGE;
            SLOUTP("%s", sqlca.sqlarm.sqlarmc);
            exit(16);
        }
        #ifdef CHECKOUT
            printf("\nAccepted ");
            for (i = 0; i < 8; i++) printf("%s", caumb.arr[i]);
        #endif
        qual_entries++;
        NEWVALS();
    }
    EXEC SQL COMMIT WORK;
    EXEC SQL CLOSE C101;
    temptabl = 2;
    break;

case 2:
    EXEC SQL DELETE FROM QUAL_CIT1;
    EXEC SQL COMMIT WORK;
    EXEC SQL OPEN C102;
    if (sqlca.sqlcode) {
        MESSAGE;
        SLOUTP("CURSOR C102");
        SLOUTP("%s", sqlca.sqlarm.sqlarmc);
        exit(16);
    }
    qual_entries = 0;
    for (;;) {
        EXEC SQL FETCH C102 INTO :caumb;
        if (sqlca.sqlcode == SQL_EOF) break;
        if (sqlca.sqlcode) {
            MESSAGE;
            SLOUTP("CURSOR C102");
            SLOUTP("%s", sqlca.sqlarm.sqlarmc);
            exit(16);
        }
        EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:caumb);
        if (sqlca.sqlcode) {
            MESSAGE;
            SLOUTP("%s", sqlca.sqlarm.sqlarmc);
            exit(16);
        }
        #ifdef CHECKOUT
            printf("\nAccepted ");
            for (i = 0; i < 8; i++) printf("%s", caumb.arr[i]);
        #endif
        NEWVALS();
        qual_entries++;
    }
    EXEC SQL COMMIT WORK;
    EXEC SQL CLOSE C102;
    temptabl = 1;
    break;

default:
    bad_temp();
}

#ifdef CHECKOUT
    printf(" %d entries in table %d", qual_entries, temptabl);
    SLOUTP("Ack");
#endif
return (int) qual_entries;
}

int querya()
/*****
 *
 * querya -- Create the subset of citations for the titles that
 *          pertain to the ANIMAL area
 *
 *****/
{
    #ifdef CHECKOUT
        printf("\nquerya ");
    #endif

```



```

SLOUT("Animal Area Search");
strout(selectorit,"Animal ");
switch (temptabl) {
  case 0:
    EXEC SQL OPEN C200;
    if (sqlca.sqlcode) {
      SLOUT("CURSOR C200");
      SENDMSG;
      SLOUT("%s", sqlca.sqlwarn.sqlwarn);
    }
    qual_entries = 0;
    for (;;) {
      EXEC SQL FETCH C200 into :aumb;
      if (sqlca.sqlcode == SQL_EOF) break;
      if (sqlca.sqlcode) {
        SLOUT("CURSOR C200");
        SENDMSG;
        SLOUT("%s", sqlca.sqlwarn.sqlwarn);
      }
      EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:aumb);
      if (sqlca.sqlcode) {
        SENDMSG;
        SLOUT("%s", sqlca.sqlwarn.sqlwarn);
      }
      qual_entries++;
    }
    EXEC SQL COMMIT WORK;
    EXEC SQL CLOSE C200;
    temptabl = 1;
    break;

  case 1:
    EXEC SQL DELETE FROM QUAL_CIT2;
    EXEC SQL COMMIT WORK;
    EXEC SQL OPEN C201;
    if (sqlca.sqlcode) {
      SLOUT("CURSOR C201");
      SENDMSG;
      SLOUT("%s", sqlca.sqlwarn.sqlwarn);
    }
    qual_entries = 0;
    for (;;) {
      EXEC SQL FETCH C201 into :aumb;
      if (sqlca.sqlcode == SQL_EOF) break;
      if (sqlca.sqlcode) {
        SENDMSG;
        SLOUT("CURSOR C201");
        SLOUT("%s", sqlca.sqlwarn.sqlwarn);
      }
      EXEC SQL INSERT INTO QUAL_CIT2 (entry_num) VALUES (:aumb);
      if (sqlca.sqlcode) {
        SENDMSG;
        SLOUT("%s", sqlca.sqlwarn.sqlwarn);
      }
      qual_entries++;
    }
    EXEC SQL COMMIT WORK;
    EXEC SQL CLOSE C201;
    temptabl = 2;
    break;

  case 2:
    EXEC SQL DELETE FROM QUAL_CIT1;
    EXEC SQL COMMIT WORK;
    EXEC SQL OPEN C202;
    if (sqlca.sqlcode) {
      SLOUT("CURSOR C202");
      SENDMSG;
      SLOUT("%s", sqlca.sqlwarn.sqlwarn);
    }
    qual_entries = 0;
    for (;;) {
      EXEC SQL FETCH C202 into :aumb;
      if (sqlca.sqlcode == SQL_EOF) break;
      if (sqlca.sqlcode) {
        SLOUT("CURSOR C202");
        SENDMSG;
        SLOUT("%s", sqlca.sqlwarn.sqlwarn);
      }
      EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:aumb);
      if (sqlca.sqlcode) {
        SENDMSG;

```

```

        SLOUTP("%s", sqlca.sqlarm.sqlarmc);
    }
    qual_entries++;
}
EXEC SQL COMMIT WORK;
EXEC SQL CLOSE C202;
temptabl = 1;
break;

default:
    bad_temp();
}

#ifdef CHECKOUT
    printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}

```

```

int querys()
/*****
 *
 *   querys -- Create the subset of citations for the titles that
 *   ===== pertain to the STRUCTURES area
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nquerys ");
#endif
    SLOUT("Structures Area Search");
    strout(scalarit, "Struct ");
    switch (temptabl) {
        case 0:
            EXEC SQL OPEN C300;
            if (sqlca.sqlcode) {
                SLOUTP("CURSOR C300");
                ENDMSG;
                SLOUTP("%s", sqlca.sqlarm.sqlarmc);
            }
            qual_entries = 0;
            for (;;) {
                EXEC SQL FETCH C300 into :cnumb;
                if (sqlca.sqlcode == SQL_EOF) break;
                if (sqlca.sqlcode) {
                    ENDMSG;
                    SLOUTP("CURSOR C300");
                    SLOUTP("%s", sqlca.sqlarm.sqlarmc);
                }
                EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:cnumb);
                if (sqlca.sqlcode) {
                    ENDMSG;
                    SLOUTP("%s", sqlca.sqlarm.sqlarmc);
                }
                qual_entries++;
            }
            EXEC SQL COMMIT WORK;
            EXEC SQL CLOSE C300;
            temptabl = 1;
            break;

        case 1:
            EXEC SQL DELETE FROM QUAL_CIT2;
            EXEC SQL COMMIT WORK;
            EXEC SQL OPEN C301;
            if (sqlca.sqlcode) {
                SLOUTP("CURSOR C301");
                ENDMSG;
                SLOUTP("%s", sqlca.sqlarm.sqlarmc);
            }
            qual_entries = 0;
            for (;;) {
                EXEC SQL FETCH C301 into :cnumb;
                if (sqlca.sqlcode == SQL_EOF) break;
                if (sqlca.sqlcode) {
                    SLOUTP("CURSOR C301");
                    ENDMSG;
                }
            }

```

```

        SLOUTP("%s", sqlca.sqlarm.sqlarmc);
    }
    EXEC SQL INSERT INTO GUAL_CIT2 (entry_num) VALUES (:cnumb);
    if (sqlca.sqlcode) {
        SENDMSG;
        SLOUTP("%s", sqlca.sqlarm.sqlarmc);
    }
    qual_entries++;
}
EXEC SQL COMMIT WORK;
EXEC SQL CLOSE C301;
temptabl = 2;
break;

case 2:
    EXEC SQL DELETE FROM GUAL_CIT1;
    EXEC SQL COMMIT WORK;
    EXEC SQL OPEN C302;
    if (sqlca.sqlcode) {
        SLOUTP("CURSOR C302");
        SENDMSG;
        SLOUTP("%s", sqlca.sqlarm.sqlarmc);
    }
    qual_entries = 0;
    for (:) {
        EXEC SQL FETCH C302 into :cnumb;
        if (sqlca.sqlcode == SQL_EOF) break;
        if (sqlca.sqlcode) {
            SLOUTP("CURSOR C302");
            SENDMSG;
            SLOUTP("%s", sqlca.sqlarm.sqlarmc);
        }
        EXEC SQL INSERT INTO GUAL_CIT1 (entry_num) VALUES (:cnumb);
        if (sqlca.sqlcode) {
            SENDMSG;
            SLOUTP("%s", sqlca.sqlarm.sqlarmc);
        }
        qual_entries++;
    }
    EXEC SQL COMMIT WORK;
    EXEC SQL CLOSE C302;
    temptabl = 1;
    break;

default:
    had_temp();
}

#ifdef CHECKOUT
    printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}

int querym()
/*****
 *
 *   querym -- Create the subset of citations for the titles that
 *   ===== pertain to the MODELING area
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nquerym ");
#endif
    SLOUT("Model Area Search");
    strout(salcrit, "Model ");

    switch (temptabl) {
        case 0:
            EXEC SQL OPEN C400;
            if (sqlca.sqlcode) {
                SLOUTP("CURSOR C400");
                SENDMSG;
                SLOUTP("%s", sqlca.sqlarm.sqlarmc);
            }

```

```

qual_entries = 0;
for (;;) {
    EXEC SQL FETCH C400 INTO :aumb;
    IF (sqlca.sqlcode = SQL_EOF) break;
    IF (sqlca.sqlcode) {
        SLOUTP("CURSOR C400");
        ENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
    EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:aumb);
    IF (sqlca.sqlcode) {
        ENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
    qual_entries++;
}
EXEC SQL COMMIT WORK;
EXEC SQL CLOSE C400;
temptabl = 1;
break;

case 1:
EXEC SQL DELETE FROM QUAL_CIT2;
EXEC SQL COMMIT WORK;
EXEC SQL OPEN C401;
IF (sqlca.sqlcode) {
    SLOUTP("CURSOR C401");
    ENDMSG;
    SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
}
qual_entries = 0;
for (;;) {
    EXEC SQL FETCH C401 INTO :aumb;
    IF (sqlca.sqlcode = SQL_EOF) break;
    IF (sqlca.sqlcode) {
        SLOUTP("CURSOR C401");
        ENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
    EXEC SQL INSERT INTO QUAL_CIT2 (entry_num) VALUES (:aumb);
    IF (sqlca.sqlcode) {
        ENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
    qual_entries++;
}
EXEC SQL COMMIT WORK;
EXEC SQL CLOSE C401;
temptabl = 2;
break;

case 2:
EXEC SQL DELETE FROM QUAL_CIT1;
EXEC SQL COMMIT WORK;
EXEC SQL OPEN C402;
IF (sqlca.sqlcode) {
    SLOUTP("CURSOR C402");
    ENDMSG;
    SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
}
qual_entries = 0;
for (;;) {
    EXEC SQL FETCH C402 INTO :aumb;
    IF (sqlca.sqlcode = SQL_EOF) break;
    IF (sqlca.sqlcode) {
        SLOUTP("CURSOR C402");
        ENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
    EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:aumb);
    IF (sqlca.sqlcode) {
        ENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
    qual_entries++;
}
EXEC SQL COMMIT WORK;
EXEC SQL CLOSE C402;
temptabl = 1;
break;

default:

```

```

        bad_tamp();
    }

#ifdef CHECKOUT
    printf(" %d entries in table %d", qual_entries, temptabl);
#endif
    return (int) qual_entries;
}

int queryw()
/*****
 *
 *   queryw -- Create the subset of citations for the author(s) whose
 *   name(s) look like the pattern given in "authorname"
 *
 *****/
{
    register int i, j;
#ifdef CHECKOUT
    printf("\nqueryw ");
#endif

    for (i = 0; i < authorname.len; i++) {
        if (authorname.arr[i] != ' ') break;
        for (j = 1; j < authorname.len; j++)
            authorname.arr[j-1] = authorname.arr[j];
        authorname.len--;
        authorname.arr[authorname.len] = '\0';
#ifdef CHECKOUT
        printf("\n%s (%u)", authorname.arr, authorname.len);
        SLOUTP("Ack");
#endif
    }
    if (!authorname.len) return (int) -1;

    strcpy(pattern.arr, authorname.arr);
    strcpy(pattern.arr, "%");
    pattern.len = strlen(pattern.arr);
    for (i = 0; i < pattern.len; i++) pattern.arr[i] = toupper(pattern.arr[i]);
    SLOUT("Author Search");
    strcpy(selector.arr, pattern.arr);

    EXEC SQL OPEN CS0s;
    if (sqlca.sqlcode) {
        SLOUTP("CURSOR CS0s");
        EXECMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
        exit(16);
    }

    qual_entries = 0;
    if (temptabl == 1) {
        EXEC SQL DELETE FROM QUAL_CIT2;
    }
    else {
        EXEC SQL DELETE FROM QUAL_CIT1;
    }
    EXEC SQL COMMIT WORK;

    EXEC SQL FETCH CS0s into :authorname;
    if (sqlca.sqlcode) {
        if (sqlca.sqlcode != SQL_EOF) {
            EXECMSG;
            SLOUTP("CURSOR CS0s");
            SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
            exit(16);
        }
    }
    else {
        do {
            #ifdef CHECKOUT
                printf("\nComparing table %d for Author number ", temptabl);
                for (i = 0; i < 5; i++) printf("%d", authorname.arr[i]);
            #endif
            switch (temptabl) {
                case 0:

```

```

EXEC SQL OPEN CS00;
if (sqlca.sqlcode) {
    SLCTF("CURSOR CS00");
    EXECMSG;
    SLCTF("%s", sqlca.sqlwarn.sqlwarn);
    exit(16);
}

for (;;) {
    EXEC SQL FETCH CS00 into :caumb;
    if (sqlca.sqlcode == SQL_EOF) break;
    if (sqlca.sqlcode) {
        SLCTF("CURSOR CS00");
        EXECMSG;
        SLCTF("%s", sqlca.sqlwarn.sqlwarn);
        exit(16);
    }
    EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:caumb);
    if (sqlca.sqlcode) {
        EXECMSG;
        SLCTF("%s", sqlca.sqlwarn.sqlwarn);
        exit(16);
    }
    #ifdef CHECKOUT
    printf("\nAccepted ");
    for (i = 0; i < 5; i++) printf("%c", caumb.arr[i]);
    #endif
    EXEC SQL COMMIT WORK;
    if (sqlca.sqlcode) {
        EXECMSG;
        SLCTF("%s", sqlca.sqlwarn.sqlwarn);
        exit(16);
    }
    qual_entries++;
    REWVALS();
}
EXEC SQL CLOSE CS00;
break;

case 1:
EXEC SQL OPEN CS01;
if (sqlca.sqlcode) {
    EXECMSG;
    SLCTF("CURSOR CS01");
    SLCTF("%s", sqlca.sqlwarn.sqlwarn);
}
for (;;) {
    EXEC SQL FETCH CS01 into :caumb;
    if (sqlca.sqlcode == SQL_EOF) break;
    if (sqlca.sqlcode) {
        SLCTF("CURSOR CS01");
        EXECMSG;
        SLCTF("%s", sqlca.sqlwarn.sqlwarn);
        exit(16);
    }
    EXEC SQL INSERT INTO QUAL_CIT2 (entry_num) VALUES (:caumb);
    if (sqlca.sqlcode) {
        EXECMSG;
        SLCTF("%s", sqlca.sqlwarn.sqlwarn);
    }
    #ifdef CHECKOUT
    printf("\nAccepted ");
    for (i = 0; i < 5; i++) printf("%c", caumb.arr[i]);
    #endif
    EXEC SQL COMMIT WORK;
    if (sqlca.sqlcode) {
        EXECMSG;
        SLCTF("%s", sqlca.sqlwarn.sqlwarn);
        exit(16);
    }
    qual_entries++;
    REWVALS();
}
EXEC SQL CLOSE CS01;
break;

case 2:
EXEC SQL OPEN CS02;
if (sqlca.sqlcode) {
    EXECMSG;
    SLCTF("CURSOR CS02");
    SLCTF("%s", sqlca.sqlwarn.sqlwarn);
    exit(16);
}

```

```

    }
    for (;;) {
        EXEC SQL FETCH CS02 INTO :camb;
        if (sqlca.sqlcode = SQL_EOF) break;
        if (sqlca.sqlcode) {
            SLOUTP("CURSOR CS02");
            SENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
            exit(16);
        }
        EXEC SQL INSERT INTO QUAL_CITI (entry_sum) VALUES (:camb);
        if (sqlca.sqlcode) {
            SENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
            exit(16);
        }
        #ifdef CHECKOUT
        printf("\nAccepted ");
        for (i = 0; i < 5; i++) printf("%s", camb.arr[i]);
        #endif
        EXEC SQL COMMIT WORK;
        if (sqlca.sqlcode) {
            SENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
            exit(16);
        }
        qual_entries++;
        NEWVALS();
    }
    EXEC SQL CLOSE CS02;
    break;

default:
    bad_temp();
}
EXEC SQL FETCH CS0s INTO :authoram;
} while (!sqlca.sqlcode);
if (sqlca.sqlcode != SQL_EOF) {
    SENDMSG;
    SLOUTP("CURSOR CS0s");
    SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    exit(16);
}
}
EXEC SQL CLOSE CS0s;
#ifdef CHECKOUT
    printf("\nold table %d", temptabl);
#endif
    temptabl = (temptabl + 2) + 1;
#ifdef CHECKOUT
    printf(" %d entries in table %d", qual_entries, temptabl);
#endif
    return (int) qual_entries;
}

int queryd()
/*****
 *
 * queryd -- Create the subset of citations for citations that were
 *          published between two years
 *
 *****/
{
    #ifdef CHECKOUT
        printf("\nqueryd ");
    #endif
    if (dates1 > dates2) {
        /* Check on the order of the dates */
        if (dates2) {
            sprintf(date1.arr, "%d", dates2);
            /* They are out of order */
            sprintf(date2.arr, "%d", dates1);
        }
        else {
            sprintf(date1.arr, "%d", dates1);
            /* Only one date entered */
            strcpy(date2.arr, "2000");
        }
    }
    else {
        if (dates2) {
            /* There is a second date specified */
            sprintf(date1.arr, "%d", dates1);
            /* They are in order */
            sprintf(date2.arr, "%d", dates2);
        }
    }
}

```

```

    else return (int) -1;          /* They are both zero! (or negative) */
}
data1.len = data2.len = 4;
strcat(salorit, "/");
strcat(salorit, data1.arr);
strcat(salorit, "/");
strcat(salorit, data2.arr);
SLOUTP("Data Search");

#ifdef CHECKOUT
printf(" Datas %s and %s", data1.arr, data2.arr);
#endif

switch (temptabl) {
case 0:
    EXEC SQL OPEN C600;
    if (sqlca.sqlcode) {
        SLOUTP("CURSOR C600");
        ENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
    }
    qual_entries = 0;
    for (;;) {
        EXEC SQL FETCH C600 INTO :aumb;
        if (sqlca.sqlcode == SQL_EOF) break;
        if (sqlca.sqlcode) {
            SLOUTP("CURSOR C600");
            ENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
        }
        EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:aumb);
        if (sqlca.sqlcode) {
            ENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
        }
        qual_entries++;
    }
    EXEC SQL COMMIT WORK;
    EXEC SQL CLOSE C600;
    temptabl = 1;
    break;

case 1:
    EXEC SQL DELETE FROM QUAL_CIT2;
    EXEC SQL COMMIT WORK;
    EXEC SQL OPEN C601;
    if (sqlca.sqlcode) {
        SLOUTP("CURSOR C601");
        ENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
    }
    qual_entries = 0;
    for (;;) {
        EXEC SQL FETCH C601 INTO :aumb;
        if (sqlca.sqlcode == SQL_EOF) break;
        if (sqlca.sqlcode) {
            SLOUTP("CURSOR C601");
            ENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
        }
        EXEC SQL INSERT INTO QUAL_CIT2 (entry_num) VALUES (:aumb);
        if (sqlca.sqlcode) {
            ENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
        }
        qual_entries++;
    }
    EXEC SQL COMMIT WORK;
    EXEC SQL CLOSE C601;
    temptabl = 2;
    break;

case 2:
    EXEC SQL DELETE FROM QUAL_CIT1;
    EXEC SQL COMMIT WORK;
    EXEC SQL OPEN C602;
    if (sqlca.sqlcode) {
        SLOUTP("CURSOR C602");
        ENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
    }
    qual_entries = 0;

```



```

    for (;;) {
        EXEC SQL FETCH C602 into :caumb;
        if (sqlca.sqlcode == SQL_EOF) break;
        if (sqlca.sqlcode) {
            SLOUTP("CURSOR C602");
            ENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
        }
        EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:caumb);
        if (sqlca.sqlcode) {
            ENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
        }
        qual_entries++;
    }
    EXEC SQL COMMIT WORK;
    EXEC SQL CLOSE C602;
    temptabl = 1;
    break;

default:
    bad_temp();
}

#ifdef CHECKOUT
    printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}

int queryt()
/*****
 *
 *   queryt -- Create the subset of citations for the titles that
 *   -----   tain the phrase given in "title_frag"
 *
 *****/
{
    register int i, j;

#ifdef CHECKOUT
    printf("\nqueryt ");
#endif
    titlefrag.len = strlen(titlefrag.arr);
    for (i = 0; i < titlefrag.len; i++) {
        if (titlefrag.arr[i] != ' ') break;
        for (j = 1; j < titlefrag.len; j++)
            titlefrag.arr[j-1] = titlefrag.arr[j];
        titlefrag.len--;
        titlefrag.arr[titlefrag.len] = '\0';
#ifdef CHECKOUT
        printf("\ntu %s", titlefrag.len, titlefrag.arr);
#endif
    }
    if (!titlefrag.len) return (int) -1;
    strcpy(pattern.arr, "%s");
    strcpy(pattern.arr, titlefrag.arr);
    strcpy(pattern.arr, "%s");
    pattern.len = strlen(pattern.arr);
    for (i = 0; i < pattern.len; i++) pattern.arr[i] = toupper(pattern.arr[i]);
    strcpy(selectorit, pattern.arr);
#ifdef CHECKOUT
    printf("\ntu %s", pattern.len, pattern.arr);
#endif
}

SLOUTP("Title Search");
switch (temptabl) {
    case 0:
        EXEC SQL OPEN C700;
        if (sqlca.sqlcode) {
            SLOUTP("CURSOR C700");
            ENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
        }
        qual_entries = 0;
        for (;;) {
            EXEC SQL FETCH C700 into :caumb;
            if (sqlca.sqlcode == SQL_EOF) break;
            if (sqlca.sqlcode) {
                SLOUTP("CURSOR C700");

```

```

        ENDMSG;
        SLOUTP("%s", sqlca.sqlarzn.sqlarzn);
    }
    EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:enumb);
    if (sqlca.sqlcode) {
        ENDMSG;
        SLOUTP("%s", sqlca.sqlarzn.sqlarzn);
    }
    qual_entries++;
}
EXEC SQL COMMIT WORK;
EXEC SQL CLOSE C700;
temptabl = 1;
break;

case 1:
EXEC SQL DELETE FROM QUAL_CIT2;
EXEC SQL COMMIT WORK;
EXEC SQL OPEN C701;
if (sqlca.sqlcode) {
    SLOUTP("CURSOR C701");
    ENDMSG;
    SLOUTP("%s", sqlca.sqlarzn.sqlarzn);
}
qual_entries = 0;
for (;;) {
    EXEC SQL FETCH C701 INTO :enumb;
    if (sqlca.sqlcode == SQL_EOF) break;
    if (sqlca.sqlcode) {
        SLOUTP("CURSOR C701");
        ENDMSG;
        SLOUTP("%s", sqlca.sqlarzn.sqlarzn);
    }
    EXEC SQL INSERT INTO QUAL_CIT2 (entry_num) VALUES (:enumb);
    if (sqlca.sqlcode) {
        ENDMSG;
        SLOUTP("%s", sqlca.sqlarzn.sqlarzn);
    }
    qual_entries++;
}
EXEC SQL COMMIT WORK;
EXEC SQL CLOSE C701;
temptabl = 2;
break;

case 2:
EXEC SQL DELETE FROM QUAL_CIT1;
EXEC SQL COMMIT WORK;
EXEC SQL OPEN C702;
if (sqlca.sqlcode) {
    ENDMSG;
    SLOUTP("CURSOR C702");
    SLOUTP("%s", sqlca.sqlarzn.sqlarzn);
}
qual_entries = 0;
for (;;) {
    EXEC SQL FETCH C702 INTO :enumb;
    if (sqlca.sqlcode == SQL_EOF) break;
    if (sqlca.sqlcode) {
        SLOUTP("CURSOR C702");
        ENDMSG;
        SLOUTP("%s", sqlca.sqlarzn.sqlarzn);
    }
    EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:enumb);
    if (sqlca.sqlcode) {
        ENDMSG;
        SLOUTP("%s", sqlca.sqlarzn.sqlarzn);
    }
    qual_entries++;
}
EXEC SQL COMMIT WORK;
EXEC SQL CLOSE C702;
temptabl = 1;
break;

default:
    bad_temp();
}

#ifdef CHECKOUT
printf(" %d entries in table %d", qual_entries, temptabl);
#endif

```

```

return (ist) qual_entries;
)

int bad_temp()
{
char msg[60];
sprintf(msg, "Bad temporary table identifier %d used!", temptabl);
SHOWMSG(msg);
exit(16);
}
/*****
*
*      query1.pc -- Routines for citation database retrieval in the
*      ----- animal area using the taxonomy table
*
*      This file contains:
*
*****/

#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>
#include <ctype.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;
VARCHAR foo[14];             /* Dummy for uniqueness fetches */

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"             /* Standard ASAN Header File */

EXEC SQL DECLARE CA001 CURSOR FOR
    SELECT anal_id, anal_name
    FROM headquarters.animal_list
    WHERE anal_name LIKE :animal;

EXEC SQL DECLARE CA002 CURSOR FOR
    SELECT anal_name, anal_id
    FROM headquarters.animal_list
    WHERE anal_id LIKE :pattern
    ORDER BY anal_id, anal_name;

EXEC SQL DECLARE CA100 CURSOR FOR
    SELECT anal_id
    FROM headquarters.animal_list
    WHERE anal_id LIKE :pattern;

EXEC SQL DECLARE CA110 CURSOR FOR
    SELECT e.entry_num
    FROM headquarters.animal_effects e
    WHERE anal_id = :nextanal;

EXEC SQL DECLARE CA111 CURSOR FOR
    SELECT e.entry_num
    FROM headquarters.animal_effects e, qual_cit1 q
    WHERE e.entry_num = q.entry_num
    AND anal_id = :nextanal;

EXEC SQL DECLARE CA112 CURSOR FOR
    SELECT e.entry_num
    FROM headquarters.animal_effects e, qual_cit2 q
    WHERE e.entry_num = q.entry_num
    AND anal_id = :nextanal;

int mkanalet()
/*****
*
*
*
*****/

```

```

*  makam1st -- Create the subset of animals that belong to the next      *
*  ----- level down in the taxonomy table                               *
*                                                                           *
*****/
{
char *clock();
FILE *txtblkf, *fopen();
int account, depth, i, lowdepth;

static char *dots[] = {" ", ". ", "... ", "... "};

#ifdef CHECKOUT
printf("\nmakam1st ");
#endif

if ( (txtblkf = fopen("txtblk\\varaniam.txt", "w")) == NULL) {
    SLOUTP("Error creating next level help window");
    sprintf(danta, "\n%s could not open animal textblock file", clock());
    return (int) DUPLICATE_OBJECT;
}

EXEC SQL OPEN CA001;                                /* Find the "object" */
if (sqlca.sqlcode) {
    SENDMSG;
    SLOUTP("CURSOR CA001");
    SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
}

account = 0;
lowdepth = 16;
for (;;) {
    EXEC SQL FETCH CA001 INTO :amal_id, :thisbeast;
    if (sqlca.sqlcode == SQL_EOF) break;
    if (sqlca.sqlcode) {
        SENDMSG;
        SLOUTP("FETCH CA001");
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
    thisbeast.arr[thisbeast.len] = '\0';

    #ifdef CHECKOUT
    amal_id.arr[amal_id.len] = '\0';
    printf("\nAnimal_id is %s", amal_id.arr, thisbeast.arr);
    #endif

    sprintf(txtblkf, "Animals below %s are:\n\n", thisbeast.arr);
    sprintf(workspace.arr, "Looking for Animals below %s ", thisbeast.arr);
    SLOUT(workspace.arr);

    for (i = 6; i > 0; i--) { /* Where are we? */
        if ((amal_id.arr[i] != '0') || (amal_id.arr[i-1] != '0')) break;
        i -= 2;
    }
    depth = i;
    if (depth < olddepth) continue;
    if (lowdepth > depth) lowdepth = depth;

    #ifdef CHECKOUT
    printf(" depth %d", depth);
    #endif

    if (depth < 6) { /* You can't go further than that */
        for (i = 0; i <= depth; i++)
            pattern.arr[i] = amal_id.arr[i];
        pattern.arr[i] = '\0';
        pattern.len = i+1;

        #ifdef CHECKOUT
        pattern.arr[pattern.len] = '\0';
        printf(" Pattern is %s", pattern.arr);
        #endif

        EXEC SQL OPEN CA002;                                /* What else is there like it */
        if (sqlca.sqlcode) {
            SENDMSG;
            SLOUTP("CURSOR CA002");
            SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
        }

        for (;;) {
            EXEC SQL FETCH CA002 INTO :anamlname, :nxtamam;
            if (sqlca.sqlcode) break;

            #ifdef CHECKOUT
            namlname.arr[namlname.len] = '\0';

```

```

        animalname.arr[animalname.len] = '\0';
        printf("\nFound %s %s", nextanimal.arr, animalname.arr);
    #endif
    if (!strcmp(nextanimal.arr, animal_id.arr, 10)) continue;

    for (i = 6; i > 0; i--) {
        if ((nextanimal.arr[i] != '0') || (nextanimal.arr[i-1] != '0')) break;
        i -= 2;
    }

    #ifdef CHECKOUT
        printf(" included");
    #endif
    animalname.arr[animalname.len] = '\0';
    sprintf(txtblkf, "%s %s\n", dots[i/2], animalname.arr);
    account++;
}
if (sqlca.sqlcode != SQL_EOF) {
    #ifdef MSG;
        sprintf(dante,
            "\n%s %s\n\t\t\t\t\t looking for animals like %s",
            clock(), sqlca.sqlwarn.sqlwarn, animal.arr);
    #endif
    sprintf(txtblkf, "%s\n",
        #ifdef SQL_CLOSE CA002;
    )
}

if (sqlca.sqlcode != SQL_EOF) {
    #ifdef MSG;
        sprintf(dante,
            "\n%s %s\n\t\t\t\t\t looking for animals like %s",
            clock(), sqlca.sqlwarn.sqlwarn, animal.arr);
    #endif
    sprintf(txtblkf, "
        == END ==\n");
    fflush(txtblkf);
    fflush(txtblkf);
    sprintf(workspace.arr, "%d entries of %d for %s", account, animals, animal.arr);
    #ifdef CHECKOUT
        #ifdef MSG;
            #endif
        #endif
        if ((lowdepth == 16) && (lowdepth > olddepth)) olddepth = lowdepth;
        if (olddepth < 8) olddepth += 2;
        return (int) account;
    }

int queryall()
/*****
 *
 *      queryall -- Find the animal of "species 1" in the taxonomy table
 *
 *
 *****/
{
    int i;

    #ifdef CHECKOUT
        printf("\nqueryall ");
    #endif

    olddepth = 0;
    animal_id.arr[0] = '\0';
    animal_id.len = 0;

    species1.len = strlen(species1.arr);
    for (i = 0; i < species1.len; i++) {
        animal.arr[i] = species1.arr[i];
        if (isspace(species1.arr[i])) break;
    }
    if (i == 0) return (int) 0; /* Animal not entered */

    animal.arr[i] = '\0'; i++;
    animal.arr[i] = '\0';
    animal.len = i;
    if (animals == MAXLIST) { /* Find the second level */
        NEW_SCREEN("akayalt");
        ADD_WINDOW("spec2entry", 5, 1);
        return (int) 0;
    }
}

int queryall2()

```

```

/*****
 *
 *   query12 -- Find the animal of "species 2" in the taxonomy table
 *
 * *****/
{
    int count, i;

#ifdef CHECKOUT
    printf("\nquery12 ");
#endif

    species2.len = strlen(species2.arr);
    for (i = 0; i < species2.len; i++) {
        animal.arr[i] = species2.arr[i];
        if (isspace(species2.arr[i])) break;
    }
    if (i != 0) { /* Animal entered ? */
        animal.arr[i] = '\0'; i++;
        animal.arr[i] = '\0';
        animal.len = i;
        if (count = skamlet()) {
            animals = count;
            REMOVE_WINDOW();
            NEW_SCREEN("akeyalt");
            ADD_WINDOW("spec2try", 6, 1);
            return (int) 0;
        }
        REMOVE_WINDOW();
        NEW_SCREEN("akeytrch");
    }
}

```

```

int query13()
/*****
 *
 *   query13 -- Find the animal of "species 3" in the taxonomy table
 *
 * *****/
{
    int count, i;

#ifdef CHECKOUT
    printf("\nquery13 ");
#endif

    species3.len = strlen(species3.arr);
    for (i = 0; i < species3.len; i++) {
        animal.arr[i] = species3.arr[i];
        if (isspace(species3.arr[i])) break;
    }
    if (i != 0) { /* Animal entered ? */
        animal.arr[i] = '\0'; i++;
        animal.arr[i] = '\0';
        animal.len = i;
        if (count = skamlet()) {
            animals = count;
            REMOVE_WINDOW();
            NEW_SCREEN("akeyalt");
            ADD_WINDOW("spec3try", 7, 1);
            return (int) 0;
        }
        REMOVE_WINDOW();
        NEW_SCREEN("akeytrch");
    }
}

```

```

int query14()
/*****
 *
 *   query14 -- Find the animal of "species 4" in the taxonomy table
 *
 * *****/
{
    int count, i;
}

```

```

#ifdef CHECKOUT
    printf("\naqueryid ");
#endif

for (i = 0; i < species4.len; i++) {
    animal.arr[i] = species3.arr[i];
    if (isspace(species4.arr[i])) break;
}
if (i != 0) {
    animal.arr[i] = '%'; i++;
    animal.arr[i] = '\0';
    animal.len = i;
    if (count = skamlet()) {
        animals = count;
    }
}
REMOVE_WINDOW();
NEW_SCREEN("akeyrah");
}

int asrch002()
/*****
 *
 *   asrch002 - Search based on entries where an Animal is selected
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nasrch002 ");
#endif
    if (queryw()) {
        NEWVALS();
        if (queryt()) {
            NEWVALS();
            if (queryd()) {
                NEWVALS();
                queryl0();
            }
        }
    }
    NEWVALS();
    if (!qual_entries) {
        SHOWMSG("No entries matching these criteria were found");
        return (int) 0;
    }
    else {
        if (!malisto()) {
            if (!shwantcit()) {
                NEW_SCREEN("citdepl");
                ADD_WINDOW("citdispection", 19, 1);
            }
        }
        return (int) 0;
    }
}

int queryl0()
/*****
 *
 *   queryl0 - Select citations that deal with specific animal(s)
 *
 *****/
{
    int i;

#ifdef CHECKOUT
    printf("\aqueryl0 ");
#endif

    if (olddepth == 0) return (int) qual_entries;
    strcpy(pattern.arr, amal_id.arr, (unsigned int) (olddepth-2));
    pattern.arr[olddepth-1] = '%';
    pattern.arr[olddepth] = '\0';
    pattern.len = strlen(pattern.arr);
    strstr(salorit, pattern.arr);

#ifdef CHECKOUT
    amal_id.arr[amal_id.len] = '\0';

```

```

    printf("\n%s, use %u characters for %s", animal_id.arr, i, pattern.arr);
#endif

    SLOUTP("Specific Animal Search");

    EXEC SQL OPEN CA100;
#ifdef CHECKOUT
    if (sqlca.sqlcode) {
        SLOUTP("CURSOR CA100");
        SENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
#endif
    qual_entries = 0;

    EXEC SQL FETCH CA100 INTO :nextanimal;
    if (sqlca.sqlcode == SQL_EOF) {
        EXEC SQL CLOSE CA100;
        return (int) qual_entries;
    }

    if (sqlca.sqlcode) {
        SENDMSG;
        SLOUTP("FETCH CA100");
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
    else {
        if (temptabl == 1) {
            EXEC SQL DELETE FROM QUAL_CIT2;
        }
        else if (temptabl == 2) {
            EXEC SQL DELETE FROM QUAL_CIT1;
        }
        EXEC SQL COMMIT WORK;
    }

#ifdef CHECKOUT
    nextanimal.arr[nextanimal.loc] = '\0';
    printf("\nTaxon %s", nextanimal.arr);
#endif

    do {
        switch (temptabl) {
            case 0:
                EXEC SQL OPEN CA110;
                if (sqlca.sqlcode) {
                    SLOUTP("CURSOR CA110");
                    SENDMSG;
                    SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
                }
                for (;;) {
                    EXEC SQL FETCH CA110 INTO :caumb;
                    if (sqlca.sqlcode == SQL_EOF) break;
                    if (sqlca.sqlcode) {
                        SLOUTP("FETCH CA110");
                        SENDMSG;
                        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
                    }
                    EXEC SQL SELECT entry_num
                        FROM qual_cit1
                        INTO :foo
                        WHERE entry_num = :caumb;
                    if (sqlca.sqlcode == SQL_EOF) {
                        EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:caumb);
                        if (sqlca.sqlcode) {
                            SLOUTP("INSERT Table 1");
                            SENDMSG;
                            SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
                        }
                        #ifdef CHECKOUT
                            caumb.arr[10] = '\0';
                            printf("\n%s qualifies", caumb.arr);
                        #endif
                        EXEC SQL COMMIT WORK;
                        qual_entries++;
                        INTERVAL();
                    }
                }
                EXEC SQL CLOSE CA110;
                break;
            case 1:
                EXEC SQL OPEN CA111;

```



```

if (sqlca.sqlcode) {
    ENDMSG;
    SLOUTP("CURSOR CALL1");
    SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
}
for (;;) {
    EXEC SQL FETCH CALL1 INTO :caumb;
    if (sqlca.sqlcode == SQL_EOF) break;
    if (sqlca.sqlcode) {
        SLOUTP("FETCH CALL1");
        ENDMSG;
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
    EXEC SQL SELECT entry_num
        FROM qual_cit2
        INTO :foo
        WHERE entry_num = :caumb;
    if (sqlca.sqlcode == SQL_EOF) {
        EXEC SQL INSERT INTO QUAL_CIT2 (entry_num) VALUES (:caumb);
        if (sqlca.sqlcode) {
            ENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
        }
        #ifdef CHECKOUT
        caumb.arr[10] = '\0';
        printf("\n%s qualifies", caumb.arr);
        #endif
        EXEC SQL COMMIT WORK;
        qual_entries++;
        NEWVALS();
    }
}
EXEC SQL CLOSE CALL1;
break;

case 2:
    EXEC SQL OPEN CALL2;
    if (sqlca.sqlcode) {
        ENDMSG;
        SLOUTP("CURSOR CALL2");
        SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }
    for (;;) {
        EXEC SQL FETCH CALL2 INTO :caumb;
        if (sqlca.sqlcode == SQL_EOF) break;
        if (sqlca.sqlcode) {
            SLOUTP("FETCH CALL2");
            ENDMSG;
            SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
        }
        EXEC SQL SELECT entry_num
            FROM qual_cit1
            INTO :foo
            WHERE entry_num = :caumb;
        if (sqlca.sqlcode == SQL_EOF) {
            EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:caumb);
            if (sqlca.sqlcode) {
                ENDMSG;
                SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
            }
            #ifdef CHECKOUT
            caumb.arr[10] = '\0';
            printf("\n%s qualifies", caumb.arr);
            #endif
            EXEC SQL COMMIT WORK;
            qual_entries++;
            NEWVALS();
        }
    }
    EXEC SQL CLOSE CALL2;
    break;

default:
    bad_temp();
}
EXEC SQL FETCH CAL00 INTO :nextanal;
} while (!sqlca.sqlcode);
if (sqlca.sqlcode != SQL_EOF) {
    ENDMSG;
    SLOUTP("FETCH CAL00");
    SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
}

```

```

temptabl = (temptabl * 2) + 1;
EXEC SQL CLOSE CA100;
#endif CHECKOUT
printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}

/*****
 *
 *      queryhl.pc -- Routines for citation database retrieval in the
 *                  human area using keyword set 1
 *
 *      This file contains:
 *
 *****/

#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asaa.h"             /* Standard ASAA Header File */

static char far *hursors[] = {

"SELECT entry_num FROM headquarters.citation_search WHERE h_annoyanc = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
WHERE s.entry_num = q.entry_num AND s.h_annoyanc = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
WHERE s.entry_num = q.entry_num AND s.h_annoyanc = 'T'",

"SELECT entry_num FROM headquarters.citation_search WHERE h_psychlgy = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
WHERE s.entry_num = q.entry_num AND s.h_psychlgy = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
WHERE s.entry_num = q.entry_num AND s.h_psychlgy = 'T'",

"SELECT entry_num FROM headquarters.citation_search WHERE h_physical = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
WHERE s.entry_num = q.entry_num AND s.h_physical = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
WHERE s.entry_num = q.entry_num AND s.h_physical = 'T'",

"SELECT entry_num FROM headquarters.citation_search WHERE h_sleep = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
WHERE s.entry_num = q.entry_num AND s.h_sleep = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
WHERE s.entry_num = q.entry_num AND s.h_sleep = 'T'",

"SELECT entry_num FROM headquarters.citation_search WHERE h_speech = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
WHERE s.entry_num = q.entry_num AND s.h_speech = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
WHERE s.entry_num = q.entry_num AND s.h_speech = 'T'",

"SELECT entry_num FROM headquarters.citation_search WHERE h_perfrmnc = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
WHERE s.entry_num = q.entry_num AND s.h_perfrmnc = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
WHERE s.entry_num = q.entry_num AND s.h_perfrmnc = 'T'"};

```

```

int hoursoffset;

int queryh1()
/*****
 *
 *   queryh1 -- Create the subset of citations for the titles that
 *   -----   pertain to the HUMAN area and also select the affect
 *               at level 1
 *
 *****/
{
    #ifdef CHECKOUT
        printf("\nqueryh1 ");
    #endif
    SLOC7("Human Area");
    switch (aff2srch[0]) {
        case 0: break;
        case 1: queryh11(); break;
        case 2: queryh12(); break;
        case 3: queryh13(); break;
        case 4: queryh14(); break;
        case 5: queryh15(); break;
        case 6: queryh16(); break;
    }
    #ifdef CHECKOUT
        default: sprintf(workspace.arr, "Invalid Human Effect %d", aff2srch[0]);
        SLOC7(workspace.arr);
    #endif
    }
    return (int) qual_entries;
}

int queryh11()
/*****
 *
 *   queryh11 -- Create the subset of citations for the titles that
 *   -----   pertain to the HUMAN area and also ANNOYANCE
 *
 *****/
{
    #ifdef CHECKOUT
        printf("\nqueryh11 ");
    #endif
    #endif
    strcat(salerit, "Human Ann");
    hoursoffset = 0;
    return (int) queryh1x();
}

int queryh12()
/*****
 *
 *   queryh12 -- Create the subset of citations for the titles that
 *   -----   pertain to the HUMAN area and also PSYCHOLOGICAL HEALTH
 *
 *****/
{
    #ifdef CHECKOUT
        printf("\nqueryh12 ");
    #endif
    #endif
    strcat(salerit, "Psychol");

    hoursoffset = 3;
    return (int) queryh1x();
}

int queryh13()
/*****
 *
 *   queryh13 -- Create the subset of citations for the titles that
 *   -----   pertain to the HUMAN area and also PHYSICAL HEALTH
 *
 *****/
{
    #ifdef CHECKOUT
        printf("\nqueryh13 ");
    #endif
    #endif
}

```

```

stroast(salorit, "Phys. Health");
hoursoffset = 6;
return (int) queryh1x();
}

```

```

int queryh14()
/*****
 *
 * queryh14 -- Create the subset of citations for the titles that
 *            pertain to the HUMAN area and also SLEEP INTERFERENCE
 *
 *****/
{
#ifdef CHECKROOT
    printf("\nqueryh14 ");
#endif
    stroast(salorit, "Human Sleep");
    hoursoffset = 9;
    return (int) queryh1x();
}

```

```

int queryh15()
/*****
 *
 * queryh15 -- Create the subset of citations for the titles that
 *            pertain to the HUMAN area and also SPEECH INTERFERENCE
 *
 *****/
{
#ifdef CHECKROOT
    printf("\nqueryh15 ");
#endif
    stroast(salorit, "Speech");
    hoursoffset = 12;
    return (int) queryh1x();
}

```

```

int queryh16()
/*****
 *
 * queryh16 -- Create the subset of citations for the titles that
 *            pertain to the HUMAN area and also TASK PERFORMANCE
 *
 *****/
{
#ifdef CHECKROOT
    printf("\nqueryh16 ");
#endif
    stroast(salorit, "Human Perf");
    hoursoffset = 18;
    return (int) queryh1x();
}

```

```

int queryh1x()
/*****
 *
 * queryh1x -- Actually do the search using hoursoffset
 *
 *****/
{
#ifdef CHECKROOT
    printf(" queryh1x %d+%d ", hoursoffset, temptabl);
#endif

    strcpy(sqlstamt.arr, hoursors[hoursoffset+temptabl]);
    sqlstamt.len = strlen(sqlstamt.arr);

    EXEC SQL PREPARE D1 FROM :sqlstamt;
    if (sqlca.sqlcode) {
        EXECMSG;
        SLOCUT("Prepare CMD");
        SLOCUT("%s", sqlca.sqlerrm.sqlrmec);
        exit(16);
    }
}

```

```

    }

EXEC SQL DECLARE CND CURSOR FOR D1;

EXEC SQL OPEN CND;
if (sqlca.sqlcode) {
    SENDMSG;
    SLOUTP("Open CND");
    SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
}

qual_entries = 0;
for (;;) {
    EXEC SQL FETCH CND into :cnumb;
    if (sqlca.sqlcode == SQL_EOF) break;
    if (sqlca.sqlcode) {
        SENDMSG;
        SLOUTP("Fetch CND");
        SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
    }

    if (temptabl == 1) {
        EXEC SQL INSERT INTO QUAL_CIT2 (entry_num) VALUES (:cnumb);
    }
    else {
        EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:cnumb);
    }
    if (sqlca.sqlcode) {
        SENDMSG;
        SLOUTP("Insert after CND");
        SLOUTP("%s", sqlca.sqlwarn.sqlwarnm);
        exit(16);
    }
    qual_entries++;
    NEWVALS();
    EXEC SQL COMMIT WORK;
}
EXEC SQL CLOSE CND;
temptabl = (temptabl + 2) + 1;

#ifdef CHECKOUT
    printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}

/*****
 *
 *      query2.pc -- Routines for citation database retrieval in the
 *      _____ human area using keyword set 2
 *
 *      This file contains:
 *
 *****/

#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"             /* Standard ASAN Header File */

static char far *h2cursors[] = {
    "SELECT d.entry_num FROM headquarters.citation_details d,\
    headquarters.citation_search s WHERE d.entry_num = s.entry_num\
    AND d.aircraft = 'T'",

    "SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
    headquarters.citation_search s WHERE s.entry_num = q.entry_num\
    AND d.entry_num = q.entry_num AND d.aircraft = 'T'",

    "SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\

```



```
int cursoroffset;
```

```

int queryh21()
/*****
*
*   queryh21 -- Create the subset of citations for the titles that
*              pertain to the HUMAN area and also Aircraft Noise
*
*****/
{
#ifdef CURSOROFF
    printf("\aqueryh21 ");
#endif
    strcat(salorit, "Aircraft");
    cursoroffset = 0;
    return (int) queryh2x();
}

```

215

```

int queryh23()
/*****
 *
 *   queryh23 -- Create the subset of citations for the titles that
 *   -----   pertain to the HUMAN area and also Seismic
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nqueryh23 ");
#endif
    strcat(salorit,"Seismic");
    cursoroffset = 6;
    return (int) queryh2x();
}

```

```

int queryh24()
/*****
 *
 *   queryh24 -- Create the subset of citations for the titles that
 *   -----   pertain to the HUMAN area and also Sonic Boom
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nqueryh24 ");
#endif
    strcat(salorit,"Boom");
    cursoroffset = 9;
    return (int) queryh2x();
}

```

```

int queryh25()
/*****
 *
 *   queryh25 -- Create the subset of citations for the titles that
 *   -----   pertain to the HUMAN area and also Terrain
 *
 *   Revision History:
 *   1.00 02/04/88 mhr      1. Creation
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nqueryh25 ");
#endif
    strcat(salorit,"Terrain");
    cursoroffset = 12;
    return (int) queryh2x();
}

```

```

int queryh26()
/*****
 *
 *   queryh26 -- Create the subset of citations for the titles that
 *   -----   pertain to the HUMAN area and also Traffic
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nqueryh26 ");
#endif
    strcat(salorit,"Traffic");
    cursoroffset = 15;
    return (int) queryh2x();
}

```

```

int queryh27()
/*****
 *
 *   queryh27 -- Create the subset of citations for the titles that
 *   -----   pertain to the HUMAN area and also Wind Noise
 *
 *****/

```



```

*****/
{
#ifdef CHECKOUT
    printf("\aqueryh27 ");
#endif
    strout(salorit, "Wind");
    cursoroffset = 10;
    return (int) queryh2x();
}

int queryh28()
/*****
 *
 *   queryh28 -- Create the subset of citations for the titles that
 *   -----   pertains to the HUMAN area and also Other Noise
 *
 *****/
{
#ifdef CHECKOUT
    printf("\aqueryh28 ");
#endif
    strout(salorit, "Other");
    cursoroffset = 21;
    return (int) queryh2x();
}

int queryh2x()
/*****
 *
 *   queryh2x -- Actually do the search using cursoroffset
 *
 *****/
{
#ifdef CHECKOUT
    printf(" queryh2x %d+%d ", cursoroffset, temptabl);
#endif

    strcpy(sqlstat.arr, h2cursors[cursoroffset+temptabl]);
    sqlstat.len = strlen(sqlstat.arr);

    EXEC SQL PREPARE D1 FROM :sqlstat;
    if (sqlca.sqlcode) {
        EXECMSG;
        SLOUTP("Prepare CMD");
        SLOUTP("%s", sqlca.sqlarm.sqlarmc);
        exit(16);
    }

    EXEC SQL DECLARE CMD CURSOR FOR D1;

    EXEC SQL OPEN CMD;
    if (sqlca.sqlcode) {
        EXECMSG;
        SLOUTP("Open CMD");
        SLOUTP("%s", sqlca.sqlarm.sqlarmc);
    }

    qual_entries = 0;
    For (;;) {
        EXEC SQL FETCH CMD into :aumb;
        if (sqlca.sqlcode == SQL_EOF) break;
        if (sqlca.sqlcode) {
            EXECMSG;
            SLOUTP("Fetch CMD");
            SLOUTP("%s", sqlca.sqlarm.sqlarmc);
        }

        if (temptabl == 1) {
            EXEC SQL INSERT INTO GOAL_CIT2 (entry_num) VALUES (:aumb);
        }
        else {
            EXEC SQL INSERT INTO GOAL_CIT1 (entry_num) VALUES (:aumb);
        }
        if (sqlca.sqlcode) {
            EXECMSG;
            SLOUTP("Insert after CMD");
        }
    }
}

```

```

        SLOWTY("%s", sqlca.sqlarnc.sqlarnc);
        exit(16);
    }
    qual_entries++;
    EXEC SQL COMMIT WORK;
    NEWVALS();
}
EXEC SQL CLOSE CMD;
temptabl = (temptabl + 2) + 1;

#ifdef CHECKOUT
    printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}
/*****
 *
 *   query3.pc -- Routines for citation database retrieval in the
 *   -----   human area using keyword set 3
 *
 *   This file contains:
 *
 *****/

#include <process.h>           /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asun.h"              /* Standard ASUN Header File */

static char far *hlcursors[] = {

"SELECT d.entry_num FROM headquarters.citation_details d,\
headquarters.citation_search s WHERE d.entry_num = s.entry_num\
AND d.field_expt = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
headquarters.citation_search s WHERE s.entry_num = q.entry_num\
AND d.entry_num = q.entry_num AND d.field_expt = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
headquarters.citation_search s WHERE s.entry_num = q.entry_num\
AND d.entry_num = q.entry_num AND d.field_expt = 'T'",

"SELECT d.entry_num FROM headquarters.citation_details d,\
headquarters.citation_search s WHERE d.entry_num = s.entry_num\
d.lab_expt = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
headquarters.citation_search s WHERE s.entry_num = q.entry_num\
AND d.entry_num = q.entry_num AND d.lab_expt = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
headquarters.citation_search s WHERE s.entry_num = q.entry_num\
AND d.entry_num = q.entry_num AND d.lab_expt = 'T'",

"SELECT entry_num FROM headquarters.citation_details d,\
headquarters.citation_search s WHERE d.entry_num = s.entry_num\
review_art = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
headquarters.citation_search s WHERE s.entry_num = q.entry_num\
AND d.entry_num = q.entry_num AND d.review_art = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
headquarters.citation_search s WHERE s.entry_num = q.entry_num\
AND d.entry_num = q.entry_num AND d.review_art = 'T'",

"SELECT entry_num FROM headquarters.citation_details d,\
headquarters.citation_search s WHERE d.entry_num = s.entry_num\
proposl_ar = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\

```

```

headquarters.citation_search s WHERE s.entry_num = q.entry_num\
AND d.entry_num = q.entry_num AND d.proposal_ar = 'T',

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
headquarters.citation_search s WHERE s.entry_num = q.entry_num\
AND d.entry_num = q.entry_num AND d.proposal_ar = 'T'";

```

```
int cursoroffset;
```

```

int queryh3()
/*****
 *
 * queryh3 -- Create the subset of citations for the titles that
 *          pertain to the HUMAN area and also select the effect
 *          at level 3
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nqueryh3 ");
#endif
    SLOCUT("Human Area");
    switch (aff2srch[2]) {
        case 0: break;
        case 1: queryh31(); break;
        case 2: queryh32(); break;
        case 3: queryh33(); break;
        case 4: queryh34(); break;
#ifdef CHECKOUT
        default: sprintf(workspace.arr, "Invalid Human Effect %d", aff2srch[2]);
                SLOCUT(workspace.arr);
#endif
    }
    return (int) qual_entries;
}

```

```

int queryh31()
/*****
 *
 * queryh31 -- Create the subset of citations for the titles that
 *            pertain to the HUMAN area and also Field Experiment
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nqueryh31 ");
#endif
    stroat(selector, "Human Fld");
    cursoroffset = 0;
    return (int) queryh3x();
}

```

```

int queryh32()
/*****
 *
 * queryh32 -- Create the subset of citations for the titles that
 *            pertain to the HUMAN area and also Lab experiment
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nqueryh32 ");
#endif
    stroat(selector, "Human Lab");
    cursoroffset = 3;
    return (int) queryh3x();
}

```

```

int queryh33()
/*****
 *
 *
 *****/

```

```

* queryh33 -- Create the subset of citations for the titles that
* -----   pertain to the HUMAN area and also Review Article
*
*****/
{
  #ifdef CHECKOUT
    printf("\nqueryh33 ");
  #endif
  strcat(salorit, "Human Rev");
  cursoroffset = 6;
  return (int) queryh3x();
}

```

```

int queryh34()
/*****
*
* queryh34 -- Create the subset of citations for the titles that
* -----   pertain to the HUMAN area and also Theoretical
*
*****/
{
  #ifdef CHECKOUT
    printf("\nqueryh34 ");
  #endif
  strcat(salorit, "Human Theor");
  cursoroffset = 9;
  return (int) queryh3x();
}

```

```

int queryh3x()
/*****
*
* queryh3x -- Actually do the search using cursoroffset
*
*****/
{
  #ifdef CHECKOUT
    printf(" queryh3x %d+%d ", cursoroffset, temptabl);
  #endif

  strcpy(sqlstmt.arr, h3cursors[cursoroffset+temptabl]);
  sqlstmt.len = strlen(sqlstmt.arr);

  EXEC SQL PREPARE D1 FROM :sqlstmt;
  if (sqlca.sqlcode) {
    ENDMSG;
    SLOUTP("Prepare CMD");
    SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    exit(16);
  }

  EXEC SQL DECLARE CMD CURSOR FOR D1;

  EXEC SQL OPEN CMD;
  if (sqlca.sqlcode) {
    ENDMSG;
    SLOUTP("Open CMD");
    SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
  }

  qual_entries = 0;
  for (;;) {
    EXEC SQL FETCH CMD into :caumb;
    if (sqlca.sqlcode == SQL_EOF) break;
    if (sqlca.sqlcode) {
      ENDMSG;
      SLOUTP("Fetch CMD");
      SLOUTP("%s", sqlca.sqlwarn.sqlwarn);
    }

    if (temptabl == 1) {
      EXEC SQL INSERT INTO QUAL_CIT2 (entry_num) VALUES (:caumb);
    }
    else {
      EXEC SQL INSERT INTO QUAL_CIT1 (entry_num) VALUES (:caumb);
    }
  }
}

```

```

    }
    if (sqlca.sqlcode) {
        EXEC SQL;
        SLOWT("Insert after CMD");
        SLOWT("%s", sqlca.sqlwarn.sqlwarn);
        exit(16);
    }
    qual_entries++;
    EXEC SQL COMMIT WORK;
    NEWVALS();
}
EXEC SQL CLOSE CMD;
temptabl = (temptabl * 2) + 1;

#ifdef CHECKOUT
    printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}
/*****
 *
 *      querypc.pc -- Routines for point of contact retrieval
 *
 *      -----
 *
 *      This file contains:
 *
 *      *****/

#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>
#include <ctype.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;
VARCHAR crit1[16];
VARCHAR crit2[46];
VARCHAR crit3[18];
VARCHAR crit4[10];
VARCHAR crit5[26];
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"             /* Standard ASAN Header File */

static int poc_type;

EXEC SQL DECLARE POCI CURSOR FOR
    SELECT first_name, last_name, title, office,
           agency_dept, st_add_div, po_box,
           misc_add, city_base, state, zipcode,
           mail_code, phone, affiliation,
           maj_attrib, min_attrib,
           area, scope_auth
    FROM superuser.point_of_contact
    WHERE UPPER(last_name) LIKE :crit1
       AND UPPER(min_attrib) LIKE :crit2
       AND UPPER(maj_attrib) LIKE :crit3
       AND UPPER(affiliation) LIKE :crit4
       AND UPPER(state) = :crit5;

EXEC SQL DECLARE POCI CURSOR FOR
    SELECT first_name, last_name, title, office,
           agency_dept, st_add_div, po_box,
           misc_add, city_base, state, zipcode,
           mail_code, phone, affiliation,
           maj_attrib, min_attrib,
           area, scope_auth
    FROM superuser.point_of_contact
    WHERE UPPER(last_name) LIKE :crit1
       AND UPPER(min_attrib) LIKE :crit2
       AND UPPER(maj_attrib) LIKE :crit3
       AND UPPER(affiliation) LIKE :crit4
       AND UPPER(city_base) LIKE :crit5;

```

```

int posrch()
/*****
 *
 * posrch -- Set up the search in the point of contact database
 *
 *
 *****/
{
    int i, j;

    static char *affil[7] = {"CITY",
                             "COUNTY",
                             "STATE",
                             "FEDERAL",
                             "MILITARY",
                             "TRIBAL",
                             ""};

#ifdef CSEARCH
    printf("\nposrch ");
#endif
    SLOWT("Point of contact search");

    strcpy(crit4.arr, affil[whichaff]);
    crit4.len = strlen(crit4.arr);

    j = strlen(contam);
    for (i = 0; i < j; i++) crit1.arr[i] = toupper(contam[i]);
    crit1.arr[j] = '\0';
    strcpy(crit1.arr, "");
    crit1.len = strlen(crit1.arr);

    j = strlen(minorat);
    for (i = 0; i < j; i++) crit2.arr[i] = toupper(minorat[i]);
    crit2.arr[j] = '\0';
    strcpy(crit2.arr, "");
    crit2.len = strlen(crit2.arr);

    j = strlen(majorat);
    for (i = 0; i < j; i++) crit3.arr[i] = toupper(majorat[i]);
    crit3.arr[j] = '\0';
    strcpy(crit3.arr, "");
    crit3.len = strlen(crit3.arr);

    j = strlen(address);
    for (i = 0; i < j; i++) crit5.arr[i] = toupper(address[i]);
    crit5.arr[j] = '\0';
    crit5.len = strlen(crit5.arr);
    if (j == 2)
        poc_type = 1;
    else {
        poc_type = 2;
        strcpy(crit5.arr, "");
        crit5.len++;
    }

    if (poc_type == 1) {
        EXEC SQL OPEN POC1;
        if (sqlca.sqlcode) {
            SLOWT("Cursor POC1");
            expMsg();
            exit(16);
        }
    }
    else {
        EXEC SQL OPEN POC2;
        if (sqlca.sqlcode) {
            SLOWT("Cursor POC2");
            expMsg();
            exit(16);
        }
    }

    if (astpoc()) {
        NEW_SCREEN("contactscreen");
        SLOWT("Sorry, but I cannot find anybody like that...");
    }
    else {
        NEW_SCREEN("pocdisplay");
    }
}

```

```

int nextpoc()
/*****
*
*   nextpoc -- Find next instance in the point of contact database
*
* *****/
{
    switch (poc_type) {
        case 1:
            EXEC SQL FETCH POC1 INTO :f_name, :l_name, :coontitle, :office,
                                     :agency_dept, :st_add_div, :po_box,
                                     :misc_add, :city_base, :state, :zipcode,
                                     :mail_code, :phone, :affiliatio,
                                     :major_attrib, :minor_attribute,
                                     :area, :scope;

            if (sqlca.sqlcode) {
                if (sqlca.sqlcode == SQL_EOF) {
                    poc_type = 0;
                    EXEC SQL CLOSE POC1;
                    SLOUTP("No more entries!");
                    return (int) SQL_EOF; }
                else {
                    SLOUTP("Fetch POC1");
                    expMsg();
                    exit(16);
                }
            }
            break;

        case 2:
            EXEC SQL FETCH POC2 INTO :f_name, :l_name, :coontitle, :office,
                                     :agency_dept, :st_add_div, :po_box,
                                     :misc_add, :city_base, :state, :zipcode,
                                     :mail_code, :phone, :affiliatio,
                                     :major_attrib, :minor_attribute,
                                     :area, :scope;

            if (sqlca.sqlcode) {
                if (sqlca.sqlcode == SQL_EOF) {
                    poc_type = 0;
                    EXEC SQL CLOSE POC2;
                    SLOUTP("No more entries!");
                    return (int) SQL_EOF; }
                else {
                    SLOUTP("Fetch POC2");
                    expMsg();
                    exit(16);
                }
            }
            break;

        case 0:
            SLOUTP("There REALLY are no more entries!");
            return (int) 0;
    }

    f_name.arr[f_name.len] = '\0';
    l_name.arr[l_name.len] = '\0';
    coontitle.arr[coontitle.len] = '\0';
    office.arr[office.len] = '\0';
    agency_dept.arr[agency_dept.len] = '\0';
    st_add_div.arr[st_add_div.len] = '\0';
    po_box.arr[po_box.len] = '\0';
    misc_add.arr[misc_add.len] = '\0';
    city_base.arr[city_base.len] = '\0';
    state.arr[state.len] = '\0';
    zipcode.arr[zipcode.len] = '\0';
    mail_code.arr[mail_code.len] = '\0';
    phone.arr[phone.len] = '\0';
    affiliatio.arr[affiliatio.len] = '\0';
    major_attrib.arr[major_attrib.len] = '\0';
    minor_attribute.arr[minor_attribute.len] = '\0';
    area.arr[area.len] = '\0';
    scope.arr[scope.len] = '\0';

    NEWVALS();
    return (int) sqlca.sqlcode;
}

int setaff(1)
/*****
*

```

```

*  nextpc -- Find next instance in the point of contact database
*  =====
*
*
*****/
int i;
{
static *affsals[] = {"CITY", "COUNTY", "STATE", "FEDERAL", "MILITARY",
                    "TRIBAL", "NONE"};

whichaf1 = i;
strcpy(affsals, affsals[i]);
NEWVALS();
}
/*****
*
*  query1.pc -- Routines for citation database retrieval in the
*  structures area using keyword set 1
*
*  This file contains:
*
*****/

```

```

#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"             /* Standard ASAN Header File */

int query1()
{
#ifdef CHECKOUT
printf("\nquery1 ");
#endif
SLCOUT("Structures Area Unavailable");
return (int) qual_entries;
}

```

```

/*****
*
*  searches.pc -- Routines that use the query routines to do searches
*  =====
*
*  This file contains:
*
*  hsrch001 - Search based on entries on first Human area screen
*  asrch001 - Search based on entries on first Animal area screen
*  sscrh001 - Search based on entries on first Structures area screen
*  mscrh001 - Search based on entries on first Modeling area screen
*
*  shwxtcit - Display the next citation on the screen
*  loadacit - Load the next citation from the database FOR DISPLAY
*
*****/

```

```

#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION; /* All SQL declarations are in */
EXEC SQL INCLUDE citvars.h;
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asan.h"             /* Standard ASAN Header File */

/*****
*
*  SQL Cursors
*
*****/

```



```

        NEW_SCREEN("citdepl");
        ADD_WINDOW("citdispection", 19, 1);
    }
    return (int) 0;
}

```

```

int herch002()
/*****
 *
 *   herch002 - Search based on entries on Human area keyword screen
 *
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nherch002 ");
#endif
    if (queryw()) {
        NEWVALS();
        if (queryt()) {
            NEWVALS();
            if (queryd()) {
                NEWVALS();
                if (queryh1()) {
                    if (queryh2()) {
                        queryh3();
                    }
                }
            }
        }
    }
    NEWVALS();
    if (!qual_entries) {
        SLOUTPR("No entries matching these criteria were found");
        return (int) 0;
    }
    else {
        if (!isolisto()) {
            if (!shwxtait()) {
                NEW_SCREEN("citdepl");
                ADD_WINDOW("citdispection", 19, 1);
            }
        }
        return (int) 0;
    }
}

```

```

int asrch001()
/*****
 *
 *   asrch001 - Search based on entries on first Animal area screen
 *
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nasrch001 ");
#endif
    if (queryw()) {
        NEWVALS();
        if (queryt()) {
            NEWVALS();
            if (queryd()) {
                NEWVALS();
                querya();
            }
        }
    }
    NEWVALS();
    if (!qual_entries) {
        SLOUTPR("No entries matching these criteria were found");
        return (int) 0;
    }
    else {
        if (!isolisto()) {
            if (!shwxtait()) {
                NEW_SCREEN("citdepl");
                ADD_WINDOW("citdispection", 19, 1);
            }
        }
        return (int) 0;
    }
}

```

```

int search001()
/*****
 *
 *      search001 - Search based on entries on first Structures area screen
 *
 *
 *****/
{
#ifdef CHECKOUT
printf("\a search001 ");
#endif

if (queryw()) {
    NEWVALS();
    if (queryt()) {
        NEWVALS();
        if (queryd()) {
            NEWVALS();
            querys();
        }
    }
}
NEWVALS();
if (!equal_entries) {
    SLOUTAP("No entries matching these criteria were found");
    return (int) 0;
}
else {
    if (!malisto()) {
        if (!shwstatit()) {
            NEW_SCREEN("citdepl");
            ADD_WINDOW("citdispection", 19, 1);
        }
    }
    return (int) 0;
}
}

```

```

int msearch001()
/*****
 *
 *      msearch001 - Search based on entries on first Modeling area screen
 *
 *
 *****/
{
#ifdef CHECKOUT
printf("\a msearch001 ");
#endif

if (queryw()) {
    NEWVALS();
    if (queryt()) {
        NEWVALS();
        if (queryd()) {
            NEWVALS();
            querys();
        }
    }
}
NEWVALS();
if (!equal_entries) {
    SLOUTAP("No entries matching these criteria were found");
    return (int) 0;
}
else {
    if (!malisto()) {
        if (!shwstatit()) {
            NEW_SCREEN("citdepl");
            ADD_WINDOW("citdispection", 19, 1);
        }
    }
    return (int) 0;
}
}

```

```

int malisto()
/*****
 *
 *      malisto - Open the cursor for main citation display screen
 *
 *
 *****/

```

```

{
#ifdef CHECKOUT
printf("\nmlisto ");
#endif

current = 0;
switch (temptabl) {

    case 1:
        EXEC SQL OPEN CD10;
        if (sqlca.sqlcode) {
            SLOCUT("Cursor CD10");
            expMsg();
            exit(16);
        }
        break;

    case 2:
        EXEC SQL OPEN CD20;
        if (sqlca.sqlcode) {
            SLOCUT("Cursor CD20");
            expMsg();
            exit(16);
        }
        break;

    default:
        bad_temp(temptabl);
}
return (int) sqlca.sqlcode;
}

int mlistc()
/*****
 * mlistc - Close the cursor for main citation display screen
 * -----
 * *****/
{
#ifdef CHECKOUT
printf("\nmlistc ");
#endif

switch (temptabl) {

    case 1:
        EXEC SQL CLOSE CD10;
        if (sqlca.sqlcode) {
            SLOCUT("Cursor CD10");
            expMsg();
            exit(16);
        }
        break;

    case 2:
        EXEC SQL CLOSE CD20;
        if (sqlca.sqlcode) {
            SLOCUT("Cursor CD20");
            expMsg();
            exit(16);
        }
        break;

    default:
        bad_temp(temptabl);
}
return (int) sqlca.sqlcode;
}

int shwnextit()
/*****
 * shwnextit - Show next citation
 * -----
 * *****/
{
int select;

#ifdef CHECKOUT
printf("\nshwnextit ");
#endif

if (select = loadnextit()) {

```

```

switch (select) {
    case SQL_EOF:

        case SQL_FETCH_OUT_OF_ORDER:
            SLOUTP("You are at the end of the list");
            REMOVE_WINDOW();
            break;

        default:
            expMsg();
            exit(16);
    }
}
else {
    current++;
    HSHVLS();
}
return (int) sqlca.sqlcode;
}

int loadnrit()
/*****
 *
 *   loadnrit - Load the citation detail display with the next citation
 *   =====
 *
 *****/
{
    int i, j, k, n;

#ifdef CHECKOUT
    printf("\nloadnrit ");
#endif

    SLOUT("Retrieving Display Text");

    if (temptabl == 1) {
        EXEC SQL FETCH CD10 INTO :cnumb, :suitable, :workspace, :datep;
        if (sqlca.sqlcode) return (int) sqlca.sqlcode;
    }
    else {
        EXEC SQL FETCH CD20 INTO :cnumb, :suitable, :workspace, :datep;
        if (sqlca.sqlcode) return (int) sqlca.sqlcode;
    }
    cnumb.arr[cnumb.len] = '\0';
    suitable.arr[1] = '\0';
    workspace.arr[workspace.len] = '\0';
    datep.arr[datep.len] = '\0';
    details_read = 0;

    j = (int) (((float) workspace.len) / ((float) 60)) + 1;
    j = j > 4 ? 4 : j;
    k = 0;
    for (i = 0; i < j; i++) {
        n = k + 60 > workspace.len ? workspace.len - k : 60;
        strcpy(entdesc[i].arr, &workspace.arr[60*i], n);
        entdesc[i].len = n;
        entdesc[i].arr[n] = '\0';
        k += n;
    }
    for (; i < 4; i++) entdesc[i].arr[0] = '\0';

    EXEC SQL OPEN CD00;
    if (sqlca.sqlcode) {
        SLOUTP("Cursor CD00");
        expMsg();
    }
    k = 0;
    for (;;) {
        v40point = &authorlist[k];
        authorlist[k].len = 40;
        /* for (i= 0; i < 40; i++) authorlist[k].arr[i] = ' '; */

        EXEC SQL FETCH CD00 INTO :authornum;
        if (sqlca.sqlcode) break;
        EXEC SQL SELECT author FROM headquarters.author_list
            INTO :v40point WHERE authornum = :authornum;
        if (sqlca.sqlcode) {
            SLOUTP("Author selection...");
            expMsg();
        }
    }
}

```

```

        exit(16);
        authorlist[k].arr[authorlist[k].len] = '\0';
        k++;
    }
    if (sqlca.sqlcode != SQL_ROW) {
        if (sqlca.sqlcode) {
            SLOUTF("Cursor CD00");
            expMsg();
            exit(16);
        }
        else SLOUTF("Next citation: more authors than screen shows");
    }
    EXEC SQL CLOSE CD00;

    for (i = k; i < 10; i++) { /* NULL for those not filled this time */
        authorlist[i].arr[0] = '\0';
        authorlist[i].len = 0;
    }
    if (k > 4) /* Truncate if we have more than 5 */
        for (i = 0; i < k; i++) {
            authorlist[k].len = i > 20 ? 20 : i;
            authorlist[k].arr[authorlist[k].len] = '\0';
        }
    return (int) 0;
}

```

```

int reserch()
/*****
 *
 *   reserch - Rescope the search. If no citations, clear the tables.
 *   =====
 *
 *****/
{
    if (!qual_entries) qsetup();
    NEW_SCREEN(oldecreen);
}

```

```

int herch010()
/*****
 *
 *   herch010 - Setup for entries on Roman keyword screen
 *   =====
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nherch010 ");
#endif
    herch000();
    if (!qual_entries) {aff2srch[0] = aff2srch[1] = aff2srch[2] = 0;}
    NEW_SCREEN("hkeyrch");
}

```

```

int herch011()
/*****
 *
 *   herch011 - Roman Effect Descriptor Type Verification
 *   =====
 *
 *****/
{
#ifdef CHECKOUT
    printf("\nherch011 ");
#endif
    for(;;) {
        if (desctype.arr[0] == 'A') {aff2srch[0] = 1; break;}
        if (desctype.arr[0] == 'P')
            if (desctype.arr[1] == 'S') {aff2srch[0] = 2; break;}
            else if (desctype.arr[1] == 'E') {aff2srch[0] = 3; break;}
        if (desctype.arr[0] == 'S')
            if (desctype.arr[1] == 'L') {aff2srch[0] = 4; break;}
            else if (desctype.arr[1] == 'P') {aff2srch[0] = 5; break;}
    }
}

```

```

        if (desctype.arr[0] == 'T') {aff2srch[0] = 6; break;}
        if ((aff2srch[0]) || (desctype.arr[0] == ' ')) {aff2srch[0] = 0; break;}
        SLOUTS("Not a valid effect");
        UPDATE_PARAM("desctype");
    }
    ADD_WINDOW("noistype", 7, 24);
    NEXT_PARAM("noistype");
}

```

```

int hsrch012()
/*****
 *      hsrch012 - Noise Descriptor Type Verification
 *
 *
 *****/
{
#ifdef CHECKOUT
printf("\nhsrch012 ");
#endif
for(;;) {
    if (desctype.arr[0] == 'A') {aff2srch[1] = 1; break;}
    if (desctype.arr[0] == 'B') {aff2srch[1] = 2; break;}
    if (desctype.arr[0] == 'S')
        if (desctype.arr[1] == 'R') {aff2srch[1] = 3; break;}
        else if (desctype.arr[1] == 'O') {aff2srch[1] = 4; break;}
    if (desctype.arr[0] == 'T')
        if (desctype.arr[1] == 'R') {aff2srch[1] = 5; break;}
        else if (desctype.arr[1] == 'R') {aff2srch[1] = 6; break;}
    if (desctype.arr[0] == 'W') {aff2srch[1] = 7; break;}
    if (desctype.arr[0] == 'O') {aff2srch[1] = 8; break;}

    if ((aff2srch[0]) || (desctype.arr[0] == ' ')) {aff2srch[1] = 0; break;}
    SLOUTS("Not a valid noise type");
    UPDATE_PARAM("noistype");
}
REMOVE_WINDOW();
ADD_WINDOW("asprtype", 7, 24);
NEXT_PARAM("asprtype");
}

```

```

int hsrch013()
/*****
 *      hsrch013 - Experimental Type Verification
 *
 *
 *****/
{
#ifdef CHECKOUT
printf("\nhsrch013 ");
#endif
for(;;) {
    if (desctype.arr[0] == 'F') {aff2srch[2] = 1; break;}
    if (desctype.arr[0] == 'L') {aff2srch[2] = 2; break;}
    if (desctype.arr[0] == 'R') {aff2srch[2] = 3; break;}
    if (desctype.arr[0] == 'T') {aff2srch[2] = 4; break;}
    if ((aff2srch[0]) || (desctype.arr[0] == ' ')) {aff2srch[2] = 0; break;}
    SLOUTS("Not valid type");
    UPDATE_PARAM("asprtype");
}
REMOVE_WINDOW();
}

```

```

int ssrch010()
/*****
 *      ssrch010 - Setup for entries on Structures keyword screen
 *
 *
 *****/
{
#ifdef CHECKOUT

```

```

printf("\nscrh010 ");
#endif

scrh000();
if (!qual_entries) {aff2scrh[0] = aff2scrh[1] = aff2scrh[2] = 0;}

NEW_SCREEN("akeyscrh");
}

int mscrh010()
/*****
*
*   mscrh010 - Setup for entries on Modeling keyword screen
*
* *****/
{
#ifdef CHECKROOT
printf("\nmscrh010 ");
#endif

scrh000();
if (!qual_entries) {aff2scrh[0] = aff2scrh[1] = aff2scrh[2] = 0;}

NEW_SCREEN("akeyscrh");
}

int ascrh010()
/*****
*
*   ascrh010 - Setup for entries on Animal keyword screen
*
* *****/
{
#ifdef CHECKROOT
printf("\nascrh010 ");
#endif

scrh000();
species1.arr[0] = '\0'; species1.len = 0;
species2.arr[0] = '\0'; species2.len = 0;
species3.arr[0] = '\0'; species3.len = 0;
species4.arr[0] = '\0'; species4.len = 0;
MSWALS();
NEW_SCREEN("akeyscrh");
}

/*****
*
*   textdisplpc -- Routines that display text fields from "MEMO_FILE"
*
*   This file contains:
*
* *****/

#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION; /* All SQL declarations are in */
EXEC SQL INCLUDE citvars.h;
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asam.h"             /* Standard ASAM Header File */

int displabet()
/*****
*
*
* *****/

```



```

*   deplabet -- Display abstract
*
*****/

{
#ifdef CHECKOUT
    printf("\n deplabet ");
#endif

if (!details_read) get_details();

if (abstract.len) {
    strcpy(nextmemo.arr, abstract.arr, 10);
    getmemo();
    NEW_SCREEN("showabs");
    ADD_WINDOW("shwastabs", 21, 1);
} else {
    SLOUTP("No abstract available");
}
}

int deplcrit()
/*****
*
*   deplcrit -- Display critique
*
*****/

{
#ifdef CHECKOUT
    printf("\n deplcrit ");
#endif

if (!details_read) get_details();

switch (critdisp) {
case 0:
    if (critique1.len == 10) {
        strcpy(nextmemo.arr, critique1.arr, 10);
        getmemo();
        NEW_SCREEN("showabs");
        ADD_WINDOW("shwastrev", 21, 1);
        NEWVALS();
        critdisp = 1;
        return (int) critdisp;
    }
case 1:
    if (critique2.len == 10) {
        strcpy(nextmemo.arr, critique2.arr, 10);
        getmemo();
        NEW_SCREEN("showabs");
        ADD_WINDOW("shwastrev", 21, 1);
        NEWVALS();
        critdisp = 2;
        return (int) critdisp;
    }
case 2:
    if (critique3.len == 10) {
        strcpy(nextmemo.arr, critique3.arr, 10);
        getmemo();
        NEW_SCREEN("showabs");
        ADD_WINDOW("shwastrev", 21, 1);
        NEWVALS();
        critdisp = 3;
        return (int) critdisp;
    }
default:
    SLOUTP("There are no more critical reviews");
    critdisp = 0;
    ADD_WINDOW("shwastabs", 21, 1);
    return (int) critdisp;
}
}

```

```

int getmemo()
/*****
 *
 *   getmemo   --   Retrieve a memofield from the database
 *
 *****/

{
    FILE *fopen(), *memo;
    int fwrite();

#ifdef CHECKOUT
    printf(" getmemo ");
#endif

    nextmemo.len = 10;

    EXEC SQL SELECT memo_text FROM headquarters.memo_file INTO :bigdisplay
        WHERE block_number = :nextmemo;
    if (sqlca.sqlcode) {
        SLOUTP("Cannot find memo block");
        RETURN;
        SLOUTP(sqlca.sqlwarn.sqlwarn);
        return (int) 0;
    }

    if (memo = fopen("tblblk\\memotxt.txt", "w") == NULL) {
        SLOUTP("Error creating memofield textblock");
        return (int) 0;
    }

    fwrite(bigdisplay.arr, bigdisplay.len, 1, memo);
    fprintf(memo, "\n\n");
    fclose(memo);
    return (int) 0;
}

int get_details()
/*****
 *
 *   get_details   --   get the pointers to the detailed memo fields
 *
 *****/

{
#ifdef CHECKOUT
    printf(" getdetails ");
#endif

    EXEC SQL SELECT ABSTRACT, CRITIQUE_R1, CRITIQUE_R2, CRITIQUE_R3
        FROM headquarters.citation_details WHERE entry_num = :camb
        INTO :abstract, :critique1, :critique2, :critique3;

    if (sqlca.sqlcode) {
        SLOUTP("Cannot find details");
        RETURN;
        SLOUTP(sqlca.sqlwarn.sqlwarn);
        return (int) 0;
    }

#ifdef CHECKOUT
    printf(" Abs %d, Crit %d, %d, %d", abstract.len, critique1.len,
        critique2.len, critique3.len);
#endif

    critdisp = 0;
    details_read = 1;
}

/*****
 *
 *   ... = ma* - All user information for a particular MFR
 *
 *****/

#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

```

```

EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;      /* All SQL declarations are in */
EXEC SQL INCLUDE hostvars.h;        /* these header files          */
EXEC SQL INCLUDE sharris.h;

EXEC SQL END DECLARE SECTION;
#include "asan.h"                    /* Standard ASAN Header File */

int mlisto()
/*****
 *
 * mlisto --- Open cursor S2 for a list of user information
 *          of navigation points on a particular MFR
 *
 * Routine executes an open cursor command for cursor S2 and then
 * returns to the calling program with the ORACLE status code.
 *
 * Note: Modifications to this function may impact the related
 *       functions mlistf() and mlists() that fetch rows and
 *       close the cursor and, possibly, functions that call
 *       these utility routines.
 *****/
{
#ifdef CHECKOUT
printf("\nmlisto ");
#endif

strcpy( cid.arr, sroid.arr);
strcpy( cid.arr, "s");
cid.len = strlen(cid.arr);

EXEC SQL DECLARE S2 CURSOR FOR
SELECT fix_label, floor_ref, ceiling_ref, fix_id, fix_type,
       artoo, fix_lat, fix_lon, fix_rad, fix_dist,
       floor, ceiling, width_left, width_right
FROM mstregments
WHERE fix_label LIKE :cid
ORDER BY fix_label;

EXEC SQL OPEN S2;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int mlistf()
/*****
 *
 * mlistf --- Fetch a row using the opened cursor S2 for MFR
 *          Navigation Point User Information
 *
 * Routine executes an fetch command for cursor S2, which is assumed
 * to have been opened, and then returns to the calling program with
 * the ORACLE status code.
 *
 * Note: Modifications to this function may impact the related
 *       functions mlisto() and mlists() that open and close the
 *       cursor and, likely, functions that call these utilities
 *****/
{
strcpy( optx3, " "); /* If you don't use VARCHAR, you have to */
strcpy( optx4, " "); /* clear the space or weird things happen */

EXEC SQL FETCH S2 INTO :a2bv, :optx1, :optx2, :curfixid,
                       :curfixtyp, :curartoo, :optx3, :optx4,
                       :curfixrad, :curfixdist, :lptr1, :lptr2,
                       :curwidleft, :curwidright;

prefixid.arr[prefixid.len] = '\0';

```

```

prefixtyp.arr[prefixtyp.lca] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int malisto()
/*****
 *
 *      malisto  ---   Close cursor S2 for Navigation Points
 *
 *
 *      Routine executes a close cursor command for cursor S2 and then
 *      returns to the calling program with the ORACLE status code.
 *
 *      Note:  Modifications to this function may impact the related
 *      functions mlistc() and mlistf() that open the cursor
 *      and fetch rows using it and, possibly, functions that
 *      call these utilities
 *
 *****/
{
#ifdef CHECKOUT
printf("\nmalisto ");
#endif

EXEC SQL CLOSE S2;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}
/*****
/*
/*      ARAH REPORT GENERATOR MODULE
/*
/*      January 26, 1987
/*
*****/

#include <process.h>          /* Header for calls to MS-DOS */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern /* Switch for header files */

EXEC SQL BEGIN DECLARE SECTION; /* All SQL declarations are in */
EXEC SQL INCLUDE hostvars.h;    /* these header files */
EXEC SQL INCLUDE faharris.h;

VARCHAR comparemis[8];          /* Last mission used */
VARCHAR comparemtr[26];         /* Last MTR used */
VARCHAR compareac[12];          /* Last aircraft used */

double exptab[27];              /* Space for "ROUTE.EAS" Table */

int day_ops[12], night_ops[12];

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE sqlca;
#include "asah.h"                /* Standard ARAH Header File */

EXEC SQL DECLARE LINDA1 CURSOR FOR
    SELECT s_label, m_ident, aircraft, activity
    FROM activities
    ORDER BY s_label, aircraft, m_ident;

EXEC SQL DECLARE LINDA2 CURSOR FOR
    SELECT day, night FROM operations
    WHERE activity = :activity
    ORDER BY month ASC;

EXEC SQL DECLARE LINDA3 CURSOR FOR
    SELECT s_label, m_ident, aircraft, activity
    FROM activities
    ORDER BY s_label, m_ident, aircraft;

```

```

char month[13][4] = {"JAN", "FEB", "MAR", "APR", "MAY", "JUN",
                     "JUL", "AUG", "SEP", "OCT", "NOV", "DEC", "ANY"};
int rta_vals1[10][2], rta_vals2[10][2], rta_vals3[10][2];
int rta_vals4[10][2], rta_vals5[10][2], total;
int annoy_flag = -2, hearing_flag = -2;
int sleep_flag = -2, livestock_flag = -2, math[20];
int speech_flag = -2, glass_flag = -2;
int compare_flag = -2, species_flag = -2, index;
double ldmr1, ldmr2, ldmr3, ldmr4, ldmr5, ldmr[20][8];
double laq1, laq2, laq3, aleval[20][5];

```

```

report(name)
char name[];
{
    FILE *fopen(), *rpt;
    int i;
    strcpy(altlev, " ");
    rpt = fopen(name, "w");
    if (rpt == NULL) {
        printf("\nbad filename %s", name);
        exit(16);
    }

    tab1a(rpt);
    tab1b(rpt);
    tab1c(rpt);
    EXEC SQL OPEN LINDAS;
    if (sqlca.sqlcode) {
        printf("Open 3: %s", sqlca.sqlwarn.sqlwarnm);
        exit(4);
    }
    for (i=0; i++) {
        EXEC SQL FETCH LINDAS INFO :srqid, :misslabl, :ac_name, :activity;
        if (sqlca.sqlcode == SQL_EOF) break;
        if (sqlca.sqlcode) {
            printf("Fetch 3: %s", sqlca.sqlwarn.sqlwarnm);
            exit(4);
        }
        srqid.arr[srqid.len] = '\0';
        misslabl.arr[misslabl.len] = '\0';
        ac_name.arr[ac_name.len] = '\0';

        fillrtvals(i);
        landuse(rpt,i);
        inconsequential(rpt,i);
        minor(rpt,i,1);
        considerable(rpt,i,1);
        notconsidered(rpt,i);
    }
    EXEC SQL CLOSE LINDAS;
    references(rpt);
    fclose(rpt);
}

```

```

report3(name, which)
char name[];
int which;
{
    FILE *fopen(), *rpt;
    int i;
    strcpy(altlev, " ");
    rpt = fopen(name, "w");
    if (rpt == NULL) {
        printf("\nbad filename %s", name);
        exit(16);
    }

    EXEC SQL OPEN LINDAS;
    if (sqlca.sqlcode) {
        printf("Open 3: %s", sqlca.sqlwarn.sqlwarnm);
        exit(4);
    }
    for (i=0; i++) {
        EXEC SQL FETCH LINDAS INFO :srqid, :misslabl, :ac_name, :activity;
        if (sqlca.sqlcode == SQL_EOF) break;
        if (sqlca.sqlcode) {
            printf("Fetch 3: %s", sqlca.sqlwarn.sqlwarnm);
            exit(4);
        }
        srqid.arr[srqid.len] = '\0';
        misslabl.arr[misslabl.len] = '\0';
        ac_name.arr[ac_name.len] = '\0';

        fillrtvals(i);
        landuse(rpt,i);
        inconsequential(rpt,i);
    }
}

```

```

        minor(xpt,1,which);
        considerable(xpt,1,which);
        notconsidered(xpt,1);
    }
    EXEC SQL CLOSE LIBDAS;
    references(xpt);
    folose(xpt);
}

tabel1(xpt)
FILE *xpt;
{
    optx3 = show.lat;
    optx4 = show.lon;
    comparemtr.arr[0] = '\0';

    EXEC SQL OPEN LIBDAS;
    if (sqlca.sqlcode) {
        printf("Cursor 1: %s", sqlca.sqlwarn.sqlwarnm);
        exit(4);
    }

    sprintf(xpt,"DESCRIPTION OF PROPOSED ACTION\n\n");
    sprintf(xpt,"The proposed action, known as %s,\n",ASSESSMENT.name);
    sprintf(xpt,"(%s)\n",ASSESSMENT.desc);
    sprintf(xpt,"consists of the use of the MFR segments as described in Table \
1.\n");
    sprintf(xpt,"The subsonic flight operations proposed for these MFR segments\n");
    sprintf(xpt,"are described in Table 2.\n\n");

    for (;;) {
        EXEC SQL FETCH LIBDAS INTO :sroid, :misslabl, :ac_name, :activity;
        if (sqlca.sqlcode == SQL_EOF) break;
        if (sqlca.sqlcode) {
            printf("Fetch 1: %s", sqlca.sqlwarn.sqlwarnm);
            exit(4);
        }

        if (!strcmp(comparemtr.arr, sroid.arr, sroid.len)) continue;
        strcpy(comparemtr.arr, sroid.arr, sroid.len);
        sroid.arr[sroid.len] = '\0';
        misslabl.arr[misslabl.len] = '\0';
        ac_name.arr[ac_name.len] = '\0';

        sprintf(xpt,"Table 1: Description of MFR %s\n\n",sroid.arr);
        sprintf(xpt,"  MAV  \n");
        sprintf(xpt,"  POINT  FIX  RAD/DIS  LATITUDE  LONGITUDE\n");
        sprintf(xpt,"-----\n");

        if (malisto()) {
            expMsg();
            exit(4);
        }

        while ( !malistf() ) {
            (int i,j;
            for (i=sroid.len, j=0; i < a2bv.len; i++, j++)
                curnavpt.arr[j] = a2bv.arr[i];
            curnavpt.arr[j] = '\0';
            curfixid.arr[curfixid.len] = '\0';

            show.lat[3] = ' ';
            show.lon[3] = ' ';
            sprintf(xpt," %s %s %3d/%3d %s %s\n",
                curnavpt.arr, curfixid.arr, curfixrad, curfixdist, show.lat, show.lon);
        }
        sprintf(xpt," \n\n");

        if (sqlca.sqlcode != SQL_EOF) printf("\n%s",sqlca.sqlwarn.sqlwarnm);
        malista();
    }
    EXEC SQL CLOSE LIBDAS;
}

tbl2(xpt)
FILE *xpt;
{
    int i;

```

```

comparamtr.arr[0] = '\0';
compareac.arr[0] = '\0';
EXEC SQL OPEN LINDA1;
if (sqlca.sqlcode) {
    printf("Cursor 1: %s", sqlca.sqlwarn.sqlwarn);
    exit(4);
}

for (;;) {
    EXEC SQL FETCH LINDA1 INTO :sruid, :misslabl, :ac_name, :activity;
    if (sqlca.sqlcode == SQL_EOF) break;
    if (sqlca.sqlcode) {
        printf("Fetch 1: %s", sqlca.sqlwarn.sqlwarn);
        exit(4);
    }

    if (strcmp(comparamtr.arr, sruid.arr, sruid.len)) {
        sruid.arr[sruid.len] = '\0';
        strcpy(comparamtr.arr, sruid.arr, sruid.len);
    }

    if (strcmp(compareac.arr, ac_name.arr, (unsigned int)12)) {
        ac_name.arr[ac_name.len] = '\0';
        strcpy(compareac.arr, ac_name.arr, (unsigned int) 12);
    }

    EXEC SQL SELECT power_units FROM ntrsalta
        INTO :pr_pwr_u WHERE aircraft = :ac_name;
    if (sqlca.sqlcode) {
        printf("Power units %s", sqlca.sqlwarn.sqlwarn);
        exit(4);
    }
    pr_pwr_u.arr[pr_pwr_u.len] = '\0';
}

misslabl.arr[misslabl.len] = '\0';

EXEC SQL SELECT sortie_size FROM missions INTO :ac_in_form
    WHERE mission = :misslabl;
if (sqlca.sqlcode) {
    printf("Sortie size %s", sqlca.sqlwarn.sqlwarn);
    exit(4);
}

EXEC SQL SELECT alt, alt_ref, pwr, spd
    FROM ntr flight_parm
    INTO :altlevel, :altlev, :ac_power, :ac_speed
    WHERE activity = :activity
    AND seq = 1;
if (sqlca.sqlcode) {
    printf("Flight parameters %s", sqlca.sqlwarn.sqlwarn);
    if (sqlca.sqlcode == SQL_EOF) { /* For checking only ! */
        altlevel = 300;
        strcpy(altlev, "?");
        ac_power = 100.00;
        ac_speed = 450;
    }
    else exit(4);
}

printf(xpt, "\n\nTable 2: Description of Flight Operations on %s \
by Month", sruid.arr);
printf(xpt, "\n\nMISSION %s (%d AIRCRAFT/FORMATION)\n\n",
    misslabl.arr, ac_in_form);
printf(xpt, "AIRCRAFT: %s POWER: %7.2lf %s\n",
    ac_name.arr, ac_power, pr_pwr_u.arr);
printf(xpt, "ALTITUDE: %6d %s SPEED: %7.2lf kts\n\n",
    altlevel, altlev, ac_speed);
printf(xpt, "
OPERATIONS\n");
printf(xpt, "MONTH DAY NIGHT\n");
printf(xpt, "-----\n");

EXEC SQL OPEN LINDA2;
if (sqlca.sqlcode) {
    printf("Cursor 2: %s", sqlca.sqlwarn.sqlwarn);
    exit(4);
}

EXEC SQL FETCH LINDA2 INTO :day_ops, :night_ops;
for (i=0; i < 12; i++) {
    printf(xpt, " %s %6d %6d\n",
        month[i], day_ops[i], night_ops[i]);
}
EXEC SQL CLOSE LINDA2;
EXEC SQL CLOSE LINDA1;
}

table3(xpt)

```

```

FILE *rpt;
{
    int    i, j;
    double temp1, temp2, add, log10();

    fprintf(rpt, "\tSUMMARY OF PREDICTED NOISE EXPOSURE\n\n");
    fprintf(rpt, "Noise exposure produced by aircraft operations may be \
specified in a variety\n");
    fprintf(rpt, "of units. The noise of low altitude high speed flights on \
Military Training\n");
    fprintf(rpt, "Routes is specified for current purposes by a cumulative noise \
metric called\n");
    fprintf(rpt, "the onset rate adjusted monthly day-night average, A-weighted \
sound level,\n");
    fprintf(rpt, "abbreviated Ldnmr. As described by Plotkin et al. (1987), \
this metric is\n");
    fprintf(rpt, "based on an integration period equal to the calendar month \
with the highest\n");
    fprintf(rpt, "number of operations.\n\n\n");

    comparestr.arr[0] = '\0';

EXEC SQL OPEN LINDA3;
if (sqlca.sqlcode) {
    printf("Cursor 3: %s", sqlca.sqlwarn.sqlwarnm);
    exit (4);
}

fprintf(rpt, "Table 3: Summary of Maximum Noise Exposure Produced by Flight \
Operations\n\n");
fprintf(rpt, " MTR    Mission    Aircraft    Month    Distance    Ldnmr\n\n");
fprintf(rpt, "-----\n\n");

for (j=0; j++) {
    EXEC SQL FETCH LINDA3 INTO :srcid, :misslabl, :ac_name, :activity;
    if (sqlca.sqlcode == SQL_ROW) break;
    if (sqlca.sqlcode) {
        printf("Fetch 3: %s", sqlca.sqlwarn.sqlwarnm);
        exit(4); }

    srcid.arr[srcid.len] = '\0';
    if (strcmp(comparestr.arr, srcid.arr, srcid.len)) {
        strcpy(comparestr.arr, srcid.arr, srcid.len);
        fprintf(rpt, "\t%-9s", srcid.arr);
        comparemis.arr[0] = '\0'; /* insure that these fail for new a/c */
        compareac.arr[0] = '\0';
    }
    else fprintf(rpt, "\n        ");

    misslabl.arr[misslabl.len] = '\0';
    if (strcmp(comparemis.arr, misslabl.arr, misslabl.len)) {
        strcpy(comparemis.arr, misslabl.arr, misslabl.len);
        comparemis.arr[comparemis.len] = '\0';
        compareac.arr[0] = '\0';

        EXEC SQL SELECT sortie_size FROM missions INTO :ac_in_form
            WHERE mission = :misslabl;

        if (sqlca.sqlcode) {
            printf("Sortie size %s", sqlca.sqlwarn.sqlwarnm);
            exit(4); }

        fprintf(rpt, "\t%-9s", misslabl.arr);
    }
    else fprintf(rpt, " ");

    ac_name.arr[ac_name.len] = '\0';
    if (strcmp(compareac.arr, ac_name.arr, ac_name.len)) {
        strcpy(compareac.arr, ac_name.arr, ac_name.len);
        compareac.arr[ac_name.len] = '\0';
        fprintf(rpt, "\t%-9s", ac_name.arr);
    }
    else fprintf(rpt, "\n        ");

EXEC SQL OPEN LINDA2;
if (sqlca.sqlcode) {
    printf("Cursor 2: activity %d\n %s", activity, sqlca.sqlwarn.sqlwarnm);
    exit (4); }

EXEC SQL FETCH LINDA2 INTO :day_ope, :night_ope;
if (sqlca.sqlcode) {
    printf("Fetch 2: %s", sqlca.sqlwarn.sqlwarnm);
    exit (4); }

```



```

EXEC SQL CLOSE LINDA2;

add = 0;
index = 0;
temp2 = 300.0;

for (i=0; i < 12; i++) {
    temp1 = (double) (day_ope[i] + 10 * night_ope[i]);
    if (temp1 > add) {
        if (temp1 < temp2) temp2 = temp1;
        add = temp1;
        index = i;
    }
}
if ((add-temp2) < 0.05) index = 12;
math[j] = index;

EXEC SQL SELECT sortia_size
FROM missions
INTO :as_in_form
WHERE mission = :misslab1;
if (sqlca.sqlcode) {
    printf("Sortia size %s", sqlca.sqlwarn.sqlwarnm);
    exit(4);
}

if (add > 0) add = 10.0 * log10(add * (double) as_in_form) - 64.1;
else add = -1000.00;

/* This is a bit sloppy we should subtract 10 LOG10 of
the number of seconds in the month instead of 64.1, but ..... */

sprintf(workspace.arr, "SELECT ACT404d FROM mtr_exp_tab", activity);
workspace.len = strlen(workspace.arr);
EXEC SQL PREPARE stmt FROM :workspace;
if (sqlca.sqlcode) {
    printf("Prepare: %s", sqlca.sqlwarn.sqlwarnm);
    exit(16);
}
EXEC SQL DECLARE D_CURS CURSOR FOR stmt;
EXEC SQL OPEN D_CURS;
if (sqlca.sqlcode) {
    printf("Dyn Open: %s", sqlca.sqlwarn.sqlwarnm);
    exit(16);
}

EXEC SQL FETCH D_CURS INTO :asptab;
if (sqlca.sqlcode) {
    printf("Exposure table: %s", sqlca.sqlwarn.sqlwarnm);
    if ((sqlca.sqlcode == SQL_EOF) ||
        (sqlca.sqlcode == NULL_FETCHED)) /* For checking only ! */
        for (i = 0; i < 27; i++) asptab[i] = 0.0;
    else exit(4);
}
EXEC SQL CLOSE D_CURS;

if (add > -500.0) {
    ldmar1 = asptab[0] + add;
    alevel[j][0] = asptab[0];
}
else {
    ldmar1 = (double) 0;
    alevel[j][0] = 0;
}

if (add > -500.0) {
    ldmar2 = asptab[16] + add;
    alevel[j][1] = asptab[16];
}
else {
    ldmar2 = (double) 0;
    alevel[j][1] = 0;
}

if (add > -500.0) {
    ldmar3 = asptab[19] + add;
    alevel[j][2] = asptab[19];
}
else {
    ldmar3 = (double) 0;
    alevel[j][2] = 0;
}

if (add > -500.0) {
    ldmar4 = 0.5*(asptab[20]+asptab[21])+add;

```



```

rta_vals[6][0] = livestock_dmg (&(rta_vals[6][1]));

rta_vals1[7][0] = speech_interference (&(rta_vals1[7][1]));
rta_vals2[7][0] = speech_interference (&(rta_vals2[7][1]));
rta_vals3[7][0] = speech_interference (&(rta_vals3[7][1]));
rta_vals4[7][0] = speech_interference (&(rta_vals4[7][1]));
rta_vals5[7][0] = speech_interference (&(rta_vals5[7][1]));

rta_vals1[8][0] = glass_breakage (&(rta_vals1[8][1]));
rta_vals2[8][0] = glass_breakage (&(rta_vals2[8][1]));
rta_vals3[8][0] = glass_breakage (&(rta_vals3[8][1]));
rta_vals4[8][0] = glass_breakage (&(rta_vals4[8][1]));
rta_vals5[8][0] = glass_breakage (&(rta_vals5[8][1]));

rta_vals1[9][0] = effects_comparison (&(rta_vals1[9][1]));
rta_vals2[9][0] = effects_comparison (&(rta_vals2[9][1]));
rta_vals3[9][0] = effects_comparison (&(rta_vals3[9][1]));
rta_vals4[9][0] = effects_comparison (&(rta_vals4[9][1]));
rta_vals5[9][0] = effects_comparison (&(rta_vals5[9][1]));
}

landuse(rpt,i)
FILE *rpt;
int i;
{
    sprintf(rpt, "\nDESCRIPTION OF LAND USE COMPATIBILITY\n\n");
    sprintf(rpt, "MFR: %s MISSION: %s AIRCRAFT: %s MONTH: %s\n\n",
        sruid.arr, misslabl.arr, ac_name.arr, month[month[i]]);
    sprintf(rpt, "Land uses compatible with the noise exposure produced by the \
flight\n");
    sprintf(rpt, "operations associated with the proposed action, as specified \
in the Joint\n");
    sprintf(rpt, "Services Land Use Planning Manual, are as noted below under \
worst case\n");
    sprintf(rpt, "assumptions. These land use interpretations are for the MFR \
segment and\n");
    sprintf(rpt, "month producing the highest noise exposure.\n");

    sprintf(rpt, "\n 0.0 miles from the MFR Centerline: \n\n");
    rta_vals1[0][0] = habitability (1, ldnmr[i][0], &(rta_vals1[0][1]), rpt);
    sprintf(rpt, "\n 0.5 miles from the MFR Centerline: \n\n");
    rta_vals2[0][0] = habitability (2, ldnmr[i][1], &(rta_vals2[0][1]), rpt);
    sprintf(rpt, "\n 1.0 miles from the MFR Centerline: \n\n");
    rta_vals3[0][0] = habitability (3, ldnmr[i][2], &(rta_vals3[0][1]), rpt);
    sprintf(rpt, "\n 1.5 miles from the MFR Centerline: \n\n");
    rta_vals4[0][0] = habitability (2, ldnmr[i][3], &(rta_vals4[0][1]), rpt);
    sprintf(rpt, "\n 2.0 miles from the MFR Centerline: \n\n");
    rta_vals5[0][0] = habitability (3, ldnmr[i][4], &(rta_vals5[0][1]), rpt);
}

incoasequential(rpt,i)
FILE *rpt;
int i;
{
    sprintf(rpt, "\nDESCRIPTION OF INCOASEQUENTIAL NOISE EFFECTS\n\n");
    sprintf(rpt, "MFR: %s MISSION: %s AIRCRAFT: %s MONTH: %s\n\n",
        sruid.arr, misslabl.arr, ac_name.arr, month[month[i]]);
    sprintf(rpt, "The following effects of noise exposure produced by the \
flight\n");
    sprintf(rpt, "operations associated with the proposed action on people, \
structures, or\n");
    sprintf(rpt, "animals were determined to be incoasequential in the current \
environmental\n");
    sprintf(rpt, "assessment:\n\n");
    check(rpt, 0);
}

minor(rpt,i, which)
FILE *rpt;
int i, which;
{
    sprintf(rpt, "\nDESCRIPTION OF NOISE EFFECTS OF MINOR IMPORTANCE\n\n");
    sprintf(rpt, "MFR: %s MISSION: %s AIRCRAFT: %s MONTH: %s\n\n",
        sruid.arr, misslabl.arr, ac_name.arr, month[month[i]]);
    sprintf(rpt, "The following effects of noise exposure produced by the \
flight\n");
    sprintf(rpt, "operations associated with the proposed action on people, \
structures, or\n");
    sprintf(rpt, "animals were determined to be of minor importance in the \
current \n");
    sprintf(rpt, "environmental assessment:\n\n");
    check(rpt, 1);
}

```

```

if (which == 1) {
    if (annoy_flag == 1) {
        bpl_annoyance(xpt);
        annoy_flag = -2;
    }
    if (hearing_flag == 1) {
        bpl_hearing(xpt);
        hearing_flag = -2;
    }
    if (sleep_flag == 1) {
        bpl_sleep(xpt);
        sleep_flag = -2;
    }
    if (livestock_flag == 1) {
        bpl_livestock(xpt);
        livestock_flag = -2;
    }
    if (speech_flag == 1) {
        bpl_speech(xpt);
        speech_flag = -2;
    }
    if (glass_flag == 1) {
        bpl_glass(xpt);
        glass_flag = -2;
    }
}
}

considerable(xpt,1,which)
FILE *xpt;
int i, which;
{
    sprintf(xpt, "\nDESCRIPTION OF NOISE EFFECTS OF CONSIDERABLE IMPORTANCE\n\n");
    sprintf(xpt, "MTR: %s MISSION: %s AIRCRAFT: %s MONTH: %s\n\n",
        sroid.arr, misslabl.arr, ac_name.arr, month[month[i]]);
    sprintf(xpt, "The following effects of noise exposure produced by the \
flight\n");
    sprintf(xpt, "operations associated with the proposed action on people, \
structures, or\n");
    sprintf(xpt, "animals were determined to be of considerable importance in \
the current\n");
    sprintf(xpt, "environmental assessment:\n\n");
    check(xpt,2);
    if (which == 1) {
        if (annoy_flag == 2) {
            bpl_annoyance(xpt);
            annoy_flag = -2;
        }
        if (hearing_flag == 2) {
            bpl_hearing(xpt);
            hearing_flag = -2;
        }
        if (sleep_flag == 2) {
            bpl_sleep(xpt);
            sleep_flag = -2;
        }
        if (livestock_flag == 2) {
            bpl_livestock(xpt);
            livestock_flag = -2;
        }
        if (speech_flag == 2) {
            bpl_speech(xpt);
            speech_flag = -2;
        }
        if (glass_flag == 2) {
            bpl_glass(xpt);
            glass_flag = -2;
        }
    }
}
}

notconsidered(xpt,1)
FILE *xpt;
int i;
{
    sprintf(xpt, "\nDESCRIPTION OF EFFECTS NOT CONSIDERED IN CURRENT \
ENVIRONMENTAL ASSESSMENT\n\n");
    sprintf(xpt, "MTR: %s MISSION: %s AIRCRAFT: %s MONTH: %s\n\n",
        sroid.arr, misslabl.arr, ac_name.arr, month[month[i]]);
    sprintf(xpt, "The following potential noise effects were not considered\
in the\n");
    sprintf(xpt, "present analyses: \n\n");

```

```

    check(xpt,-1);
    sprintf(xpt, "\n\nReasons that these potential effects were not considered\
included\n");
    sprintf(xpt, "insufficient information for evaluation, insufficient \
precision of \n");
    sprintf(xpt, "estimation of noise exposure, and lack of generally accepted \
means of \n");
    sprintf(xpt, "producing quantitative estimates of magnitudes of potential \
effects.\n");
}

references(xpt)
FILE *xpt;
{
    sprintf(xpt, "\n\nREFERENCES\n\n");
    sprintf(xpt, "Busnel, R. 1978. \"Introduction,\" in J. L. Fletcher and \
R. G. Busnel (eds), \n");
    sprintf(xpt, "Effects of Noise on Wildlife. Academic Press, New York.\n\n");
    sprintf(xpt, "Cottareau, P. 1972. Sonic boom exposure effects: effects on \
animals.\n");
    sprintf(xpt, "Journal of Sound Vibration 20 (4):531-534.\n\n");
    sprintf(xpt, "Environmental Protection Agency. 1980. Guidelines for noise \
impact\n");
    sprintf(xpt, "analysis. Office of Air, Noise, and Radiation, United States \
Environmental\n");
    sprintf(xpt, "Protection Agency (USEPA).\n\n");
    sprintf(xpt, "Environmental Protection Agency. 1974. Information on levels \
of\n");
    sprintf(xpt, "environmental noise requisite to protect public health and \
welfare with an\n");
    sprintf(xpt, "adequate margin of safety. EPA 550/9 74 004.\n\n");
    sprintf(xpt, "Fletcher, J. L., and Busnel, R. G., eds. 1978. Effects of \
noise\n");
    sprintf(xpt, "on wildlife. Academic Press.\n\n");
    sprintf(xpt, "Hershaw, R. L., and Higgins, T. E., eds. 1973. Statistical \
prediction model\n");
    sprintf(xpt, "for glass breakage from nominal sonic booms. Federal \
Aviation\n");
    sprintf(xpt, "Administration Report FAA RD 73-79.\n\n");
    sprintf(xpt, "Kinshaw, W. R.; Ball, W. B.; Ladsen, T. A.; McNeil, E. C. E.; \
and Taylor, J.\n");
    sprintf(xpt, "P. 1970. An annotated bibliography on animal response to sonic \
booms and\n");
    sprintf(xpt, "other loud sounds. Washington, D. C.\n\n");
    sprintf(xpt, "International Civil Aviation Organization (ICAO). 1970. Sonic \
boom effects\n");
    sprintf(xpt, "on the Animal Kingdom. Sonic Boom Panel, Montreal, 12 21 \
October 1970.\n\n");
    sprintf(xpt, "Newman, J. and K. P. Beattie. 1965. Aviation Noise Effects. \
FAA\n");
    sprintf(xpt, "RM 65 2. Federal Aviation Administration, Noise Abatement \
Branch.\n");
    sprintf(xpt, "Washington, D. C.\n\n");
    sprintf(xpt, "Wilson, G. W.; Mills, H. K.; Sommer, H. C.; and Guild, H. \
1969. \
\"Sonic Booms\n");
    sprintf(xpt, "resulting from extremely low altitude supersonic flight: \
measurements and\n");
    sprintf(xpt, "observations on houses, livestock and people.\" Aerospace \
Medical Research\n");
    sprintf(xpt, "Laboratories, Wright-Patterson Air Force Base, Report AMRL TR \
68 52.\n\n");
    sprintf(xpt, "Plotkin, Kenneth J.; Sutherland, Louis C.; and Molino, John A.\n");
    sprintf(xpt, "1967. \"Environmental Noise Assessment for Military Aircraft\n");
    sprintf(xpt, "Training Routes. Volume 2: Recommended Noise Matrix.\" \
Aerospace\n");
    sprintf(xpt, "Medical Research Laboratories, Wright-Patterson Air Force \
Base,\n");
    sprintf(xpt, "Report AMRL-TR-67-001.\n\n");
    sprintf(xpt, "Shotton, L. R. 1982. \"Response of Wildlife and Farm Animals \
to Low Level\n");
    sprintf(xpt, "Military Jet Overflight.\" The Reporter II (6):161-164.\n");
}

bpl_livestock(xpt)
FILE *xpt;
{
    sprintf(xpt, "\n\nLIVESTOCK DAMAGE\n\n");
    sprintf(xpt, "The U.S. Environmental Protection Agency (EPA) has reviewed \
the literature\n");
    sprintf(xpt, "on noise effects in domestic animals (Dufour 1980). In \
general, there is an\n");
}

```

```

    sprintf(rpt, "overall trend for domestic animals to adapt to intermittent \
    (aircraft or\n");
    sprintf(rpt, "aircraft-like noise under 120 dB (decibels). Buznal (1978)\
    reviewed\n");
    sprintf(rpt, "effects around large airports and found no evidence to \
    indicate noise-\n");
    sprintf(rpt, "related adverse effects.\n\n");
    sprintf(rpt, "Negative behavioral effects from exposure to sonic booms are\
    rare among\n");
    sprintf(rpt, "domestic animals such as horses, cattle, sheep and poultry \
    (Cottareau 1972;\n");
    sprintf(rpt, "Fletcher & Buznal 1978; Hineshaw et al. 1970; Hinson et al. \
    1968;\n");
    sprintf(rpt, "International Civil Aviation Organization [ICAO] 1970). Large\
    farm animals\n");
    sprintf(rpt, "may respond with spontaneous activity (i.e. galloping, \
    bellowing, jumping).\n");
    sprintf(rpt, "Poultry show mild reactions to the booms in most cases, but in\
    less than tami\n");
    sprintf(rpt, "percent of the cases chickens reacted with crowding, covering,\
    or noise.\n");
    sprintf(rpt, "There was reported to be no significant effect on egg \
    production, milk\n");
    sprintf(rpt, "production, or food consumption. Pigs, both in the open and \
    in shelter,\n");
    sprintf(rpt, "show a tendency to be quiet (ICAO 1970). Observations show \
    greater\n");
    sprintf(rpt, "responses resulting from low-level subsonic flights, \
    motorcycles, paper\n");
    sprintf(rpt, "blown by the wind and other startling stimuli (ICAO 1970). \
    There appears to\n");
    sprintf(rpt, "be no report of panic, injury or negative effects upon \
    reproductive success\n");
    sprintf(rpt, "(Fletcher & Buznal 1978).\n\n");

```

}

bpl_hearing(rpt)

FILE *rpt;

{

```

    sprintf(rpt, "\n\nHEARING DAMAGE RISK\n\n");
    sprintf(rpt, "Hearing loss can be either temporary or permanent. A \
    noise-induced\n");
    sprintf(rpt, "temporary threshold shift is a temporary loss of hearing \
    experienced after a\n");
    sprintf(rpt, "relatively short exposure to excessive noise. A noise-induced\
    threshold\n");
    sprintf(rpt, "shift means that the detection level of sound has been \
    increased. Recovery\n");
    sprintf(rpt, "is fairly rapid after cessation of the noise. A noise-induced\
    permanent\n");
    sprintf(rpt, "threshold shift is an irreversible loss of hearing caused by \
    prolonged\n");
    sprintf(rpt, "exposure to excessive noise. This loss is essentially \
    indistinguishable\n");
    sprintf(rpt, "from the normal hearing loss associated with aging. Permanent\
    hearing loss\n");
    sprintf(rpt, "is generally associated with destruction of the hair cells of \
    the inner ear.\n");
    sprintf(rpt, "Based on EPA (Environmental Protection Agency) criteria, \
    hearing loss is not\n");
    sprintf(rpt, "expected for people living within noise contours below DNL \
    levels of 75 dB\n");
    sprintf(rpt, "(decibels). Further, as stated in the EPA \"Levels Document,\" \
    changes in\n");
    sprintf(rpt, "hearing levels of <5 dB are generally not considered noticeable\
    or\n");
    sprintf(rpt, "significant (EPA 1974).\n\n");

```

}

bpl_speech(rpt)

FILE *rpt;

{

```

    sprintf(rpt, "\n\nSPEECH INTERFERENCE\n\n");
    sprintf(rpt, "Speech interference associated with aircraft noise is a \
    primary source of\n");
    sprintf(rpt, "annoyance to individuals on the ground. The disruption of \
    leisure\n");
    sprintf(rpt, "activities (such as listening to the radio, television, music,\

```

```

and\n");
    sprintf(xpt, "conversation) gives rise to frustration and irritation. \
Quality speech\n");
    sprintf(xpt, "communication is obviously also important in the classroom, \
office, and\n");
    sprintf(xpt, "industrial settings. Researchers have found that aircraft \
noise of 75 dB\n");
    sprintf(xpt, "(decibels) annoyed the highest percentage of the population \
when it\n");
    sprintf(xpt, "interfered with the television sound. Eighty percent of the \
list of\n");
    sprintf(xpt, "annoyances for the surveyed population was flickering of the \
television\n");
    sprintf(xpt, "picture and interference with casual conversation by aircraft \
noise (Herman\n");
    sprintf(xpt, "% Beattie 1985).\n\n");
}

hpl_annoyance(xpt)
FILE *xpt;
{
    sprintf(xpt, "\n\nNOISE ANNOYANCE\n\n");
    sprintf(xpt, "Noise annoyance is defined by the U.S. Environmental \
Protection Agency (EPA)\n");
    sprintf(xpt, "as any negative subjective reaction to noise on the part of an \
individual or\n");
    sprintf(xpt, "group (EPA 1978). \"Except in the case of speech interference, \
however, the\n");
    sprintf(xpt, "degree of interference is hard to specify and difficult to \
relate to the\n");
    sprintf(xpt, "level of noise exposure\" (EPA 1978). \"Aircraft noise may \
... be found\n");
    sprintf(xpt, "annoying because it may startle people, cause houses to shake, \
or elicit\n");
    sprintf(xpt, "fear of a crash\" (EPA 1978).\n\n");
}

hpl_glass(xpt)
FILE *xpt;
{
    sprintf(xpt, "\n\nSTRUCTURAL DAMAGE\n\n");
    sprintf(xpt, "By far, the largest percentage of sonic boom damage claims \
stem from broken\n");
    sprintf(xpt, "or cracked glass. All of the tests conducted in the United \
States have\n");
    sprintf(xpt, "confirmed that glass damage is the most prevalent damage caused \
by sonic\n");
    sprintf(xpt, "booms (Marshay & Higgins 1973). Because the microstructure of \
glass is\n");
    sprintf(xpt, "amorphous rather than crystalline, the practical design \
strength of glass\n");
    sprintf(xpt, "is dependent on the surface scratch condition. Glass that has \
been\n");
    sprintf(xpt, "sandblasted, scratched, or nicked will not exhibit the same \
strength as\n");
    sprintf(xpt, "a properly installed relatively new pane of glass.\n\n");
}

hpl_sleep(xpt)
FILE *xpt;
{
    sprintf(xpt, "\n\nSLEEP INTERFERENCE\n\n");
    sprintf(xpt, "Sleep interference is one of the factors contributing to \
aircraft noise\n");
    sprintf(xpt, "annoyance. Airport nighttime restrictions have been employed \
to minimize\n");
    sprintf(xpt, "this annoyance. In the case of nighttime operations, an \
anterior maximum\n");
    sprintf(xpt, "sound level (AIm) of 72 dB (decibels) is identified as an \
acceptable sleep\n");
    sprintf(xpt, "interference condition for a windows-closed condition. This \
corresponds to\n");
    sprintf(xpt, "an interior AIm of about 55 dB.\n\n");
    sprintf(xpt, "To provide a basis for estimation of the number of people who \
could be\n");
    sprintf(xpt, "awakened by a specific noise event, data developed by \

```

```

Goldstein and Lukas\n");
    sprintf(rpt, "(1980) were used to develop a relationship between the ASEL \
value and the\n");
    sprintf(rpt, "Percent of exposed persons who would be awakened by the noise \
event. These\n");
    sprintf(rpt, "data indicated that the percent awakened by a specific \
interior noise level\n");
    sprintf(rpt, "can be expressed by the following equation:\n\n");
    sprintf(rpt, "Percent Awakened = 1.1(ASEL) - 49.5\n");
    sprintf(rpt, "where ASEL = the interior A-weighted sound exposure level.\n\n");
    sprintf(rpt, "Since noise must penetrate the home to disturb sleep, interior \
noise levels\n");
    sprintf(rpt, "will be lower than outside levels due to the absorption of \
sound energy\n");
    sprintf(rpt, "attenuation by the structure. The amount of attenuation \
provided by the\n");
    sprintf(rpt, "building is dependent on the type of construction and whether \
windows are\n");
    sprintf(rpt, "open or closed. The Environmental Protection Agency recommends \
attenuation\n");
    sprintf(rpt, "factors of 17 dB (decibels) for summertime (windows open) \
residential\n");
    sprintf(rpt, "conditions and 27 dB for wintertime (windows closed) \
conditions.\n");
    sprintf(rpt, "Incorporating the attenuation factors into the above equation \
gives the\n");
    sprintf(rpt, "following relationships for the percent awakened under \
summertime and\n");
    sprintf(rpt, "wintertime conditions:\n\n");
    sprintf(rpt, "Percent Awakened (summer) = 1.1(ASEL - 17) - 49.5\n");
    sprintf(rpt, "Percent Awakened (winter) = 1.1(ASEL - 27) - 49.5\n");
    sprintf(rpt, "Percent Awakened (winter) = 1.1(ASEL) - 79.2\n");
}

check(rpt, which)
FILE *rpt;
int which;
{
    sprintf(rpt, "\n 0.0 miles from the MTR Centerline: \n\n");
    if (rtn_vals1[1][1] == which) {
        sprintf(rpt, "Prevalence of Annoyance Among Population\n");
        annoy_flag = which;
    }
    if (rtn_vals1[2][1] == which) {
        sprintf(rpt, "Hearing Damage Risk to Residential Population\n");
        hearing_flag = which;
    }
    if (rtn_vals1[3][1] == which) {
        sprintf(rpt, "Sleep Interference of Residential Population\n");
        sleep_flag = which;
    }
    if (rtn_vals1[5][1] == which) {
        sprintf(rpt, "Reproductive Success or Population Size of an Endangered \
Species\n");
        species_flag = which;
    }
    if (rtn_vals1[6][1] == which) {
        sprintf(rpt, "Economic Damage to Livestock\n");
        livestock_flag = which;
    }
    if (rtn_vals1[7][1] == which) {
        sprintf(rpt, "Speech Interference of Residential Population\n");
        speech_flag = which;
    }
    if (rtn_vals1[8][1] == which) {
        sprintf(rpt, "Glass Breakage Claims Among Residential Population\n");
        glass_flag = which;
    }
    if (rtn_vals1[9][1] == which) {
        sprintf(rpt, "Effects Comparison Module\n");
        compare_flag = which;
    }
    if (annoy_flag != which && hearing_flag != which && sleep_flag != which &&
        livestock_flag != which && speech_flag != which && glass_flag != which &&
        compare_flag != which && species_flag != which)
        sprintf(rpt, "There were no effects of noise in this category at this \
distance\n");

    sprintf(rpt, "\n 0.5 miles from the MTR Centerline: \n\n");
    if (rtn_vals2[1][1] == which) {

```



```

        sprintf(rpt, "Prevalence of Annoyance Among Population\n");
        annoy_flag = which;
    }
    if (rta_vals2[2][1] == which) {
        sprintf(rpt, "Hearing Damage Risk to Residential Population\n");
        hearing_flag = which;
    }
    if (rta_vals2[3][1] == which) {
        sprintf(rpt, "Sleep Interference of Residential Population\n");
        sleep_flag = which;
    }
    if (rta_vals2[5][1] == which) {
        sprintf(rpt, "Reproductive Success or Population Size of an Endangered \
Species\n");
        species_flag = which;
    }
    if (rta_vals2[6][1] == which) {
        sprintf(rpt, "Economic Damage to Livestock\n");
        livestock_flag = which;
    }
    if (rta_vals2[7][1] == which) {
        sprintf(rpt, "Speech Interference of Residential Population\n");
        speech_flag = which;
    }
    if (rta_vals2[8][1] == which) {
        sprintf(rpt, "Glass Breakage Claims Among Residential Population\n");
        glass_flag = which;
    }
    if (rta_vals2[9][1] == which) {
        sprintf(rpt, "Effects Comparison Module\n");
        compare_flag = which;
    }
    if (annoy_flag != which && hearing_flag != which && sleep_flag != which &&
        livestock_flag != which && speech_flag != which && glass_flag != which &&
        compare_flag != which && species_flag != which)
        sprintf(rpt, "There were no effects of noise in this category at this \
distance\n");

    sprintf(rpt, "\n  1.0 miles from the MTR Centerline: \n\n");
    if (rta_vals3[1][1] == which) {
        sprintf(rpt, "Prevalence of Annoyance Among Population\n");
        annoy_flag = which;
    }
    if (rta_vals3[2][1] == which) {
        sprintf(rpt, "Hearing Damage Risk to Residential Population\n");
        hearing_flag = which;
    }
    if (rta_vals3[3][1] == which) {
        sprintf(rpt, "Sleep Interference of Residential Population\n");
        sleep_flag = which;
    }
    if (rta_vals3[5][1] == which) {
        sprintf(rpt, "Reproductive Success or Population Size of an Endangered \
Species\n");
        species_flag = which;
    }
    if (rta_vals3[6][1] == which) {
        sprintf(rpt, "Economic Damage to Livestock\n");
        livestock_flag = which;
    }
    if (rta_vals3[7][1] == which) {
        sprintf(rpt, "Speech Interference of Residential Population\n");
        speech_flag = which;
    }
    if (rta_vals3[8][1] == which) {
        sprintf(rpt, "Glass Breakage Claims Among Residential Population\n");
        glass_flag = which;
    }
    if (rta_vals3[9][1] == which) {
        sprintf(rpt, "Effects Comparison Module\n");
        compare_flag = which;
    }
    if (annoy_flag != which && hearing_flag != which && sleep_flag != which &&
        livestock_flag != which && speech_flag != which && glass_flag != which &&
        compare_flag != which && species_flag != which)
        sprintf(rpt, "There were no effects of noise in this category at this \
distance\n");

    sprintf(rpt, "\n  1.5 miles from the MTR Centerline: \n\n");
    if (rta_vals4[1][1] == which) {
        sprintf(rpt, "Prevalence of Annoyance Among Population\n");
        annoy_flag = which;
    }

```

```

    }
    if (rta_vals4[2][1] == which) {
        sprintf(rpt, "Hearing Damage Risk to Residential Population\n");
        hearing_flag = which;
    }
    if (rta_vals4[3][1] == which) {
        sprintf(rpt, "Sleep Interference of Residential Population\n");
        sleep_flag = which;
    }
    if (rta_vals4[5][1] == which) {
        sprintf(rpt, "Reproductive Success or Population Size of an Endangered \
Species\n");
        species_flag = which;
    }
    if (rta_vals4[6][1] == which) {
        sprintf(rpt, "Economic Damage to Livestock\n");
        livestock_flag = which;
    }
    if (rta_vals4[7][1] == which) {
        sprintf(rpt, "Speech Interference of Residential Population\n");
        speech_flag = which;
    }
    if (rta_vals4[8][1] == which) {
        sprintf(rpt, "Glass Breakage Claims Among Residential Population\n");
        glass_flag = which;
    }
    if (rta_vals4[9][1] == which) {
        sprintf(rpt, "Effects Comparison Module\n");
        compare_flag = which;
    }
    if (annoy_flag != which && hearing_flag != which && sleep_flag != which &&
        livestock_flag != which && speech_flag != which && glass_flag != which &&
        compare_flag != which && species_flag != which)
        sprintf(rpt, "There were no effects of noise in this category at this \
distance\n");

    sprintf(rpt, "\n 2.0 miles from the MTR Centerline: \n\n");
    if (rta_vals5[1][1] == which) {
        sprintf(rpt, "Prevalence of Annoyance Among Population\n");
        annoy_flag = which;
    }
    if (rta_vals5[2][1] == which) {
        sprintf(rpt, "Hearing Damage Risk to Residential Population\n");
        hearing_flag = which;
    }
    if (rta_vals5[3][1] == which) {
        sprintf(rpt, "Sleep Interference of Residential Population\n");
        sleep_flag = which;
    }
    if (rta_vals5[5][1] == which) {
        sprintf(rpt, "Reproductive Success or Population Size of an Endangered \
Species\n");
        species_flag = which;
    }
    if (rta_vals5[6][1] == which) {
        sprintf(rpt, "Economic Damage to Livestock\n");
        livestock_flag = which;
    }
    if (rta_vals5[7][1] == which) {
        sprintf(rpt, "Speech Interference of Residential Population\n");
        speech_flag = which;
    }
    if (rta_vals5[8][1] == which) {
        sprintf(rpt, "Glass Breakage Claims Among Residential Population\n");
        glass_flag = which;
    }
    if (rta_vals5[9][1] == which) {
        sprintf(rpt, "Effects Comparison Module\n");
        compare_flag = which;
    }
    if (annoy_flag != which && hearing_flag != which && sleep_flag != which &&
        livestock_flag != which && speech_flag != which && glass_flag != which &&
        compare_flag != which && species_flag != which)
        sprintf(rpt, "There were no effects of noise in this category at this \
distance\n");
}
/*****
/*
/*      rptasn.pc -- ASAW Main Program (Temporary driver for the
/*                      report generator
/*
*/

```



```

else {
    j = (sizeof ASSESSMENT.desc) - 1;
    if (workspace.len >= j) {
        for(i = 0; i < j; i++) ASSESSMENT.desc[i] = workspace.arr[i];
        ASSESSMENT.desc[j] = '\0';
    } else strcpy(ASSESSMENT.desc, workspace.arr);
}

if (argc > 2)
    strcpy(targv, argv[2]);
else
    strcpy(targv, "pra");

Vinit();
printf("%c[24;1H", 27);

exit(2);
}

callrpt()
{
    report(targv);
}

callrpt3()
{
    report3(targv, 0);
}

callrpt4()
{
    report3(targv, 1);
}

```

B.1 C Language Source Code

```

/* ASAW Effects Modules */
/* */
/* Created: January 8, 1987 */
/* */

/* The return or success codes returned by the procedures in this */
/* module have the following meanings: */
/* -2 = execution is not possible because the module is */
/* not implemented yet */
/* -1 = execution is not possible because the available */
/* input is incomplete or improper */
/* 0 = the precision of the estimate is satisfactory */
/* 1 = the module executed, but the precision of the */
/* estimate is unsatisfactory */
/* */

/* The severity of affect codes returned by the procedures in this */
/* module have the following meanings: */
/* -1 = effect not considered in the current analysis */
/* 0 = magnitude of predicted effect is inconsequential */
/* 1 = magnitude of predicted effect is of minor importance */
/* 2 = magnitude of predicted effect is of considerable */
/* importance */
/* */

#include "stdio.h"

int table[20][5] = {0, 0, 0, 30, 25, 0, 0, 35, 30, 25, 0, 0, 35, 30,
                    25, 0, 0, 0, 30, 25, 40, 35, 30, 25, 1, 0, 0, 30,
                    25, 1, 0, 0, 0, 30, 25, 0, 0, 30, 25, 1, 0, 0, 0,
                    0, 0, 0, 0, 30, 25, 1, 35, 30, 1, 1, 1, 0, 35, 30,
                    25, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
                    1, 0, 30, 25, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
                    0, 0, 0, 1, 1, 1, 1, 1, 1};

char table2[20][80] = { "Family housing", "Bachelor housing",
                        "Transient lodging - hotels, motels",
                        "Classrooms, libraries, churches",
                        "Offices and administration buildings, military",
                        "Offices - business and professional",

```

```

"Hospitals, medical facilities, nursing homes",
"Dental clinic, medical dispensaries",
"Outdoor music shells",
"Retail stores, restaurants, banks, movie theaters",
"Flight line operations, maintenance and training",
"Industrial, manufacturing and laboratories",
"Outdoor sports arenas, outdoor spectator sports",
"Playgrounds, active sport recreational areas",
"Neighborhood parks", "Gymnasiums, indoor pools",
"Outdoor - frequent speech communication",
"Outdoor - unfrequent speech communication",
"Livestock farming, animal breeding",
"Agricultural (except livestock)";

```

```

habitability (which, ldn, effectcode, rpt)
double ldn;
int *effectcode, which;
FILE *rpt;

```

```

/* The above tables and below look-up computations were derived from
the Joint Services Noise Planning Manual as referenced in Table 6.3
of Theodore J. Schultz's book, Community Noise Rating, Second Edition
(1962). */

```

```

{
    int i, index;
    char name[15];

    if (ldn > 65 && ldn <= 69) {
        index = 4;
        *effectcode = 1;
    }
    else if (ldn <= 74) {
        index = 3;
        *effectcode = 2;
    }
    else if (ldn <= 79) {
        index = 2;
        *effectcode = 2;
    }
    else if (ldn <= 84) {
        index = 1;
        *effectcode = 2;
    }
    else if (ldn <= 89) {
        index = 0;
        *effectcode = 2;
    }
}

if (ldn < 65) {
    fprintf(rpt, " The following are compatible land uses: \n");
    for (i = 0; i < 20; i++)
        fprintf(rpt, " %s \n", table2[i]);
    *effectcode = 0;
}

else if (ldn > 89) {
    fprintf(rpt, " The following are incompatible land uses: \n");
    for (i = 0; i < 20; i++)
        fprintf(rpt, " %s \n", table2[i]);
    *effectcode = 2;
}

else {
    fprintf(rpt, " The following are compatible land uses: \n");
    for (i = 0; i < 20; i++)
        if (table[i][index] == 1)
            fprintf(rpt, " %s \n", table2[i]);

    fprintf(rpt,
        "\n The following land uses are compatible with interior noise\n");
    fprintf(rpt, " reduction noted in parentheses:\n");
    for (i = 0; i < 20; i++)
        if (table[i][index] != 1 && table[i][index] != 0)
            fprintf(rpt, " %s \n", table2[i], table[i][index]);
    fprintf(rpt, "\n The following are incompatible land uses: \n");
    for (i = 0; i < 20; i++)
        if (table[i][index] == 0)
            fprintf(rpt, " %s \n", table2[i]);
}

return(0);
}

```

```

annoyance (ldn, affectcode)
double ldn;
int    *affectcode;

/* The below calculation of percent of the population that will be highly
   annoyed by a sound of level ldn was copied from Theodore J. Schultz's
   article, "Synthesis of social surveys on noise annoyance", published in
   the Journal of the Acoustical Society of America, Vol. 64, No. 2, August
   1978, pp 377-405. */

{
    double parha, pow();

    if (ldn < 48 || ldn > 85) {
        *affectcode = -1;
        return(-1);
    }
    else {
        parha = 0.8553 * ldn - 0.0401 * pow(ldn,2.) + 0.00047 * pow(ldn,3.);
        if (parha < 5)
            *affectcode = 0;
        else if (parha <= 10)
            *affectcode = 1;
        else if (parha <= 20)
            *affectcode = 2;
        else
            *affectcode = 2;
        return(0);
    }
}

hearing_damage (alevel, affectcode)
double alevel;
int    *affectcode;

/* The calculation of ea was derived from William Burns' book, Noise
   and Man, Second Edition (1973), page 238. The dB loss values were
   read from the 50t curve on the bottom graph on page 239 of the same
   book */

{
    int ea, temp; /* ea = A-weighted noise immission level */

    ea = (int) alevel + 10; /* 10 is derived from 10*log(duration of sound
                           exposure in years) which is assumed to be 5
                           years here */

    temp = ea + 5; /* round to the nearest 5 dBs */

    if (temp <= 2.5)
        ea = temp;
    else
        ea += (5 - temp);

    /* ea of 95 -> 50% of the population would experience a 1.2dB hearing loss */
    /* 90 -> 2.3dB loss, 95 -> 4.5dB loss, 100 -> 8.7dB loss */
    /* 105 -> 14.1dB loss, 110 -> 22.3dB loss, 115 -> 31.2dB loss */
    /* 120 -> 40.0dB loss, 125 -> 43.1dB loss, 130 -> 43.1dB loss */

    if (ea <= 95)
        *affectcode = 0;
    else if (ea <= 110)
        *affectcode = 1;
    else
        *affectcode = 2;
    return(0);
}

sleep_interference (alevel, affectcode, i)
double alevel;
int    *affectcode, i;

{
    double per_awakened;

    if (i > 4 && i < 10) /* SUMMER */
        per_awakened = 1.1 * alevel - 68.2;
    else /* WINTER */
        per_awakened = 1.1 * alevel - 79.2;

    if (per_awakened <= 5.0)

```

```

        *affectcode = 0;
    else if (par_awakened <= 10.0)
        *affectcode = 1;
    else
        *affectcode = 2;
    return(0);
}

sem_land_use (affectcode)
int *affectcode;

{
    *affectcode = -1;
    return(-2);
}

endangered_species_reproduction (affectcode)
int *affectcode;

{
    *affectcode = -1;
    return(-2);
}

livestock_dmg (affectcode)
int *affectcode;

{
    *affectcode = -1;
    return(-2);
}

speech_interference (affectcode)
int *affectcode;

{
    *affectcode = -1;
    return(-2);
}

glass_breakage (affectcode)
int *affectcode;

{
    *affectcode = -1;
    return(-2);
}

effects_comparison (affectcode)
int *affectcode;

{
    *affectcode = -1;
    return(-2);
}
}

/*****
 *
 *      cu.c -- ASAN GRASS color utilities.
 *
 *      THIS CODE ASSUMES 256-COLOR LOT AND 2 OVERLAY PLANES.
 *
 *****/

#include "grass.h"
#include "gylabs.h"

init_colors ()
{
    int j;

    /* ??? ( C_CURSOR , 255 , 255 , 255 ) */

    InitCI( C_HNS      , 0 , 0 , 0 );

    InitCI( C_SCR_FRAME, 255 , 255 , 255 );
}

```

```

InitCI( C_WIN_FRAME, 255, 255, 255 );
InitCI( C_LOS_FRAME, 255, 255, 255 );
InitCI( C_LOS_TEXT, 255, 255, 255 );

InitCI( C_BLACK, 1, 1, 1 );
InitCI( C_RED, 255, 0, 0 );
InitCI( C_GREEN, 0, 255, 0 );
InitCI( C_BLUE, 0, 0, 255 );
InitCI( C_YELLOW, 255, 255, 0 );
InitCI( C_MAGENTA, 255, 0, 255 );
InitCI( C_CYAN, 0, 255, 255 );
InitCI( C_WHITE, 255, 255, 255 );

/* for DMA elevations, use a smooth palette at low intensity */
ig_palet( C_1ST_DMA, NUM_DMA_C, 0.25, 0.50 );

/* for categories, use a repeating smooth palette at high intensity */
for ( j = C_1ST_CAT; j <= MAX_C - C_CAT_REPEAT + 1; j += C_CAT_REPEAT )
    ig_palet( j, C_CAT_REPEAT, 0.50, 0.50 );
if ( j < MAX_C )
    ig_palet( j, MAX_C - j + 1, 0.50, 0.50 );

next_cat_pv = C_1ST_CAT; /* initialize pival to use for category */
}

ig_palet ( spv, nclrs, i, s ) /* load smooth palette in Imagraph LUT */
int spv; /* first LUT location to load */
int nclrs; /* how many LUT locations to load */
double i, s; /* intensity, saturation */
{
    double hinc, h, r, g, b;
    int pv;

    if ( ( spv < 0 ) || ( nclrs < 1 ) || ( spv + nclrs - 1 > MAX_C ) )
    {
        ERROR("ig_palet: bad parameter");
        return( -1 );
    }

    hinc = 360.0 / (double) nclrs;
    h = 0;

    for ( pv = spv; pv < spv + nclrs; pv++ )
    {
        hisrgb( h, i, s, &r, &g, &b );
        InitCI( pv, dround(r * 255.0), dround(g * 255.0), dround(b * 255.0) );
        h += hinc;
    }
}

/*
 * hisrgb() -- converts (h,i,s) color to (r,g,b) intensities
 *
 * call: hisrgb( h, i, s, &r, &g, &b );
 *
 * calling args (normalized before use):
 * 0.0 <= h <= 360.0 (red = 120.0, green = 240.0, blue = 0.0)
 * 0.0 <= i <= 1.0 0.0 <= s <= 1.0
 * returned args:
 * 0.0 <= r <= 1.0 0.0 <= g <= 1.0 0.0 <= b <= 1.0
 */

hisrgb ( h, i, s, r, g, b )
double h, i, s, *r, *g, *b;
{
    double m, m2;

    /* normalize calling args */

    while ( h < 0.0 ) h += 360.0;
    while ( h > 360.0 ) h -= 360.0;
    while ( i < 0.0 ) i += 1.0;
    while ( i > 1.0 ) i -= 1.0;
    while ( s < 0.0 ) s += 1.0;
    while ( s > 1.0 ) s -= 1.0;

    if ( i > 0.5 ) m = i + s - ( i * s );
    else m = i * ( 1.0 + s );

```



```

m2 = ( 2.0 * i ) - m;

if ( h < 60.0 ) *r = m2 + ( m - m2 ) * ( h / 60.0 );
else if ( h < 180.0 ) *r = m;
else if ( h < 240.0 ) *r = m2 + ( m - m2 ) * ( ( 240.0 - h ) / 60.0 );
else *r = m2;

if ( h < 120.0 ) *g = m2;
else if ( h < 180.0 ) *g = m2 + ( m - m2 ) * ( ( h - 120.0 ) / 60.0 );
else if ( h < 300.0 ) *g = m;
else *g = m2 + ( m - m2 ) * ( ( 360.0 - h ) / 60.0 );

if ( h < 60.0 ) *b = m;
else if ( h < 120.0 ) *b = m2 + ( m - m2 ) * ( ( 120.0 - h ) / 60.0 );
else if ( h < 240.0 ) *b = m2;
else if ( h < 300.0 ) *b = m2 + ( m - m2 ) * ( ( h - 240.0 ) / 60.0 );
else *b = m; /* questionable */
}

/* end of m2.c */
/*****
 *
 *      dmain.c -- dummy asan mainline to test map screen features
 *
 *****/

#define DS(str)

/*
 * needs data for where's my finger on map.
 *
 * show coordinates of specified location on map.
 */

#include <stdio.h>
#include "asantype.h"
#include "asan.h"

/* DEFINED IN GRAPHICS CODE TO BE SUPPLIED BY FME: */

double shmidt;
extern char dname[12];

/* ASSUMED TO BE DEFINED ELSEWHERE IN ASAN USER CODE: */

COORDINATE ent;
COORDINATE show;
ASANHEADR ASSESSMENT;
char idptofint[6];
int shwareaLW;
int shwareaLW1;
int shwareaPST;
int shwareaPST1;

extern int HSA_DEBUG_FEATURES; /* for debug */

main ()
{
    Ginit(); /* initialize graphics */

    DS( strcpy( dname[0], "FirstMap" ); )
    DS( strcpy( dname[7], "JustaMap" ); )
    DS( strcpy( dname[14], "Last_Map" ); )

    HSA_DEBUG_FEATURES = 1; /* enable U debugging features */
    Uinit(); /* disappear into U */
}

dummy ()
{
    SLOUTP( "**** inside dummy() ****" );
}

lnt2dec ( ent )
int ent;
{
    SLOUTP( "**** inside lnt2dec() ****" );
}

```

```

loaddec ( ent )
int ent;
{
    SLOUTP ("*** inside loaddec() ***");
}

AAAHoccm ( arr )
unsigned char *arr;
{
    SLOUTP ("*** inside AAAHoccm() ***");
}

/* end of dasan.c */
/*****
 *
 *      da.c -- AAAH GRASS cell file handlers.
 *
 *****/

#define PUTRASTOR 0          /* = 1 if okay to use PutRast */

#define DB(str)

#include "grass.h"
#include "globals.h"

#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>

extern char    DMA_alav_being_drawn;

static char    fname[80];
static int     fh;
static int     nbytes;
static int     scurrow;
static char    xbuf[MAXCOLS];
static char    ybuf[MAXCOLS];

d_cell ()
{
    int r, c;
    int oc, dcw; /* drawing-method-test only */

    set_display_params();          /* set up window & viewport */

    sprintf( fname, "%s\\%s\\%s\\cell\\%s",
        gisdbase, mapset, location, layer2add);

    if ( ( fh = open( fname, O_RDONLY | O_BINARY ) ) == -1 )
    {
        SLOUTP( "d_cell: open %s failed", fname );
        return( -1 );
    }

    scurrow = 0;

    # if PUTRASTOR
        /* color is wired in cell file */
    # else
        /* major kludge alert */
    # endif /* PUTRASTOR */

    clip_to_window();

    for ( r = 0 ; r < W_rows ; r++ )
    {
        if ( ( nbytes = read( fh, xbuf, W_ncols ) ) != W_ncols )
        {
            DCERR( "%s out of data at row %d", fname, r );
            return( -1 );
        }

        # if PUTRASTOR

            if ( DMA_alav_being_drawn )          /* fiddle with colors */
                for ( oc = 0 ; oc < W_ncols ; oc++ )
                    if ( xbuf[oc] != 0 )
                        xbuf[oc] = ( (int) ( (double) xbuf[oc] / 17. ) + 1. );

```

```

        PutRast( W_1, W_t - sarrow, W_1 + W_ncols, xbuf );

    #   else

        for ( cc = 0; cc < W_ncols; cc++ )
            if ( xbuf[cc] != 0 )
            {
                if ( DMA_alav_being_drawn ) /* fiddle with colors */
                    SetFG( (int) ( (double) xbuf[cc] / 17. ) + 1. );
                else
                    SetFG( xbuf[cc] );

                xline( W_1 + cc, W_t - sarrow,
                    W_1 + cc + 1, W_t - sarrow + 1, 0x40 );
            }

    #   endif /* PUTRASTOR */

        sarrow++;
    }

    #   ifdef DOWT_CARE
        if ( ( nbytes = read( fh, xbuf, W_ncols ) ) > 0 )
        {
            DCHRR( "fs has data left over (%d bytes)", fh, nbytes );
            return( -1 );
        }
    #   endif /* DOWT_CARE */

        close( fh );
        clip_to_screen();
        return( 0 );
    }

screen2cell ()
{
    SLOUTP( "screen2cell() is not yet available" );
}

static DCHRR ( a, b, c, d, e, f, g )
{
    char msgbuf[80];
    char tmpbuf[80];

    sprintf( tmpbuf, a, b, c, d, e, f, g );
    sprintf( msgbuf, "%d cell: " );
    strcat( msgbuf, tmpbuf );
    SLOUTP( msgbuf );
    close( fh );
    clip_to_screen();
}

/* end of da.c */
/*****
 *
 *   dd.c -- ADAM GRASS code to draw a "digit" file on the screen.
 *
 *****/

#define DS(str)

#include "grass.h"
#include "gylobe.h"

#include <font1.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>

static char    fnam[80];
static FILE    *dfp;
static char    xbuf[100];
static int     fline;
static char    b_type;
static int     count;

/* block type */

d_digit ()
{

```

```

int i, j, x, y;
int last_sig; /* index to latest non-whitespace char (-1=none) */

set_display_params(); /* set up window & viewport */

sprintf( fname, "%s\\%s\\%s\\digit\\%s",
         gisdbase, mapset, location, layer2add);

if ( ( dfp = fopen( fname, "r" ) ) == NULL )
{
    SLPRINTF( "?? digit: fopen %s failed", fname );
    return( -1 );
}

fline = 0;

/* skip the 14 lines of unwanted header stuff */
for ( i = 0 ; i < 14 ; i++ )
{
    fline++;
    if ( fgets( xbuf, 100, dfp ) == NULL )
    {
        DERR( "fgets error" );
        return( -1 );
    }
}

clip_to_window();

/*
 * **** SLUDGE ALERT:
 *
 * We know that current digit files contain exactly one category each,
 * and that next_cat_pv has just been incremented in build_legend(), so
 * the following color selection code is MUCH simpler than it ought to be.
 *
 * Color/legend stuff will be VERY screwed up if any of the subsequent
 * error returns is taken.
 */

SetPG( next_cat_pv - 1 );

/* process the digit file one block at a time */
while ( ( b_type = fgets( dfp ) ) != EOF )
{
    DS( SLPRINTF( "dd: 1st char of block = %c = %c", b_type, b_type ); )
    switch( b_type )
    {
        case 'A': /* polyline */
            fline++;
            if ( fgets( dfp ) != ':' )
            {
                DERR( "bad char follows block type" );
                return( -1 );
            }
            DS( SLPRINTF( "dd A: got ':'" ); )
            if ( fscanf( dfp, "%d", &count ) < 1 )
            {
                DERR( "bad count or early EOF" );
                return( -1 );
            }
            DS( SLPRINTF( "dd A: got count=%d", count ); )
            fline++;
            if ( fscanf( dfp, "%d %d", &dx1, &dy1 ) < 2 )
            {
                DERR( "bad 1st vertex or early EOF" );
                return( -1 );
            }
            DS( printf( "dd A: got 1st vertex %d %d\n", dx1, dy1 ); )
            for ( i = 0 ; i < count - 1 ; i++ )
            {
                fline++;
                if ( fscanf( dfp, "%d %d", &dx2, &dy2 ) < 2 )
                {
                    DERR( "bad data or early EOF" );
                    return( -1 );
                }
            }
            DS( printf( "dd A: got next vertex %d %d, call d_line()\n", dx2, dy2 ); )
            d_line();
            dx1 = dx2;

```



```

        clip_to_screen();
    }

/* end of dd.c */
/*****
 *
 *          draw.c -- code to draw actual graphics
 *
 *****/

#define DS(str)

#include "grass.h"
#include "gglobe.h"

#define XPIX(xx)      ( W_l + dround( ( xx - V_west ) / UTM_per_pixal ) )
#define YPIX(yy)      ( W_b + dround( ( yy - V_south ) / UTM_per_pixal ) )

extern int      xfont, yfont;          /* Inagrapht text params */
extern int      chardepth, charwidth;

static int      t_size = 1;           /* text zoom factor (1-16) */
static int      xpix, ypix;

d_line ()
{
    /* draw a line */
    aline( XPIX( dx1 ) , YPIX( dy1 ) , XPIX( dx2 ) , YPIX( dy2 ) , 0x40 );
}

d_text ( ctype )
char ctype;
/* SCREEN_COORDS or UTM_COORDS */
{
    int i, xf, yf;

    if ( ctype == UTM_COORDS )
    {
        xpix = XPIX( dx1 );
        ypix = YPIX( dy1 );
    }
    else /* SCREEN_COORDS */
    {
        xpix = ix1;
        ypix = iy1;
    }

    t_size = 1;

    if ( font == 'R' )
        xf = xpix - ( strlen( txt ) * charwidth );
    else if ( font == 'C' )
        xf = xpix + ( ( strlen( txt ) * charwidth ) / 2 );
    else /* assume font == 'L' */
        xf = xpix;

    yf = ypix - ( chardepth / 2 ); /* supplied Y is vertical center */

    for ( i = 0 ; i < strlen( txt ) ; i++ )
    {
        xfont = xf;
        yfont = yf;
        DS(SLOWTY("gputa: xfont=%d yfont=%d char=%04o",xfont,yfont,txt[i],txt[i]));
        if ((txt[i] == ' ') || (txt[i] == '\t')) /* space or tab -> space */
            xf += charwidth;
        else if ( txt[i] < 040 ) /* ignore ctrl chars altogether */
            ;
        else /* send everything else to screen */
        {
            gputa( txt[i] , t_size , 0 );
            xf += charwidth;
        }
    }

    SetData( -0 ); /* restore solid pattern */
}

/* end of draw.c */
/*****
 *
 *****/

```

```

*      gglobe.c -- globals and parameters for ASAN CGRASS.      *
*      *      *      *      *      *      *      *      *      *      *
*      ***** WARNING: THIS FILE MUST MATCH gglobe.h *****
*      *      *      *      *      *      *      *      *      *
*****/

#include "grass.h"

char  dname[MAX_PLAYERS][12]; /* displayed layer names */
int   num_layers;             /* how many layers now displayed */
LINK  leg[MAX_CATS];          /* legend/cat data structure */
int   num_llines;             /* how many legend/cat data entries now */
int   next_cat_pv;            /* next pival to use for category */

char  *aspect;                /* current aspect ("ASAN") */
char  *location;              /* current location name ("Salls") */
char  *gisbase;               /* current map data base dir ("grass\maps") */
char  vname[12];              /* current viewport name ("C"=coarse, etc.) */
char  wname[12];              /* current window name */
char  layer2add[12];          /* name of layer to add to display */
char  layer2del[12];          /* name of layer to remove from display */
char  layersavename[12];      /* name to use for layer being saved */
char  laytype;                /* layer is: 'C'= cell, 'D'= digit, 'L'= DLS */

char  window_frame_drawn;     /* YES if window frame drawn, else NO */
char  legend_drawn;           /* YES if legend drawn, else NO */
char  DTM_elev_drawn;         /* YES if DTM elevations are drawn, else NO */

double dx1, dx2, dy1, dy2;    /* some vertices */
int     ix1, ix2, iy1, iy2;    /* some vertices */
char    txt[100];              /* data needed to draw text */
double  height, width, rotation, slant;
int     font;
char    just;

int     W_t = SCR_T;           /* current window, in screen coords */
int     W_b = SCR_B;
int     W_l = SCR_L;
int     W_r = SCR_R;
int     W_rows;                /* how many rows */
int     W_cols;                /* how many columns */

int     V_proj;                /* current viewport, in VMS */
int     V_scase;
double  V_north;
double  V_south;
double  V_west;
double  V_east;
double  V_ns_res;
double  V_ew_res;
int     V_format;

double  VMS_per_pival;

/* end of gglobe.c */
/*****
*
*      grass.c -- ASAN graphics main routines.
*
*****/

#define DB(str)

#include "grass.h"
#include "gglobe.h"

char    DTM_elev_being_drawn;

static char dflt_aspect[] = { 'a', 's', 'a', 'n', '\0' };
static char dflt_location[] = { 'S', 'a', 'l', 'l', 's', '\0' };
static char dflt_gisbase[] = { '\\', 'g', 'r', 'a', 's', 's', '\\', 'm', 'a', 'p', 's', '\0' };
static char dflt_wname[] = { 'w', 'i', 'n', 'd', 'o', 'w', '\0' };

static char fname[80];
static FILE *fp;
static int  after_Vinit;
static char del_layer_in_progress; /* KLUDGE ALERT */

```

```

init ()
/* call me first */
{
    aspect = (char *) dflt_aspect; /* used to be environment vars ... */
    location = (char *) dflt_location;
    gisdbase = (char *) dflt_gisdbase;
    strcpy( vname , dflt_vname );

    begin_is_graphics();
    /* load default_palette(); */
    InitFont();

    init_colors();

    after_Vinit = NO;
    clear_screen();
    after_Vinit = YES;
    DMA_alav_being_drawn = NO;
}

new_view ( vstr ) /* change viewport */
char *vstr;
{
    if ( strcmp( vstr , vname ) == 0 ) /* ignore if no change */
        return;

    if ( { strcmp( vstr , "Sells.C" ) == 0 } ||
          { strcmp( vstr , "Ajo.M" ) == 0 } ||
          { strcmp( vstr , "Sells.M" ) == 0 } ||
          { strcmp( vstr , "Ajo.F" ) == 0 } ||
          { strcmp( vstr , "Sells.F" ) == 0 } )
    {
        strcpy( vname , vstr );
        NEWVALS();
        clear_screen();
    }
    else
        SLOUTS( "? unknown view ts" , vstr );
}

clear_screen () /* erase screen & clear parameters */
{
    int i;

    erase_graphics_display();

    for ( i = 0 ; i < MAX_LAYERS ; i++ ) /* no layers displayed */
        strcpy( dname[i] , "" );
    num_layers = 0;

    if ( after_Vinit ) /* NEWVALS before Vinit -> bizarre crashes */
        NEWVALS();
}

erase_graphics_display () /* just erase the screen */
{
    clip_to_screen();
    clr( C_RGB ); /* clear screen to background color */
    SetFG( C_SCR_FRAME ); /* draw screen frame */
    arect( SCR_L + 1 , SCR_B + 1 , SCR_R - 1 , SCR_T - 1 , 0x40 );

    window_frame_drawn = NO; /* window frame is gone */
    legend_drawn = NO; /* legend is gone */
    DMA_alav_drawn = NO; /* DMA elevations are gone */
}

add_layer () /* add a layer to display */
{
    int i;

    if ( strcmp( vname , "" ) == 0 ) /* if no active view, give up */
    {
        SLOUTS( "choose a view first" );
        strcpy( layer2add , "" );
        NEWVALS();
        return( -1 );
    }
}

```



```

for ( i = 0 ; i < strlen( layer2add ) ; i++ ) /* assure name is uppercase */
    if ( ( layer2add[i] >= 'a' ) && ( layer2add[i] <= 'z' ) )
        layer2add[i] -= 040;

if ( strcmp( layer2add , "DRA_MLEV" ) == 0 ) /* special case */
{
    draw_dra_mlev();
    return( 0 );
}

/* see if it exists */

laytype = ' ';
sprintf( fnam, "%s\\%s\\%s\\digit\\%s",          /* digit file? */
        gisdbase, mapset, location, layer2add);
if ( ( fp = fopen( fnam , "r" ) ) != NULL )
    laytype = 'D';
if ( laytype == ' ' )
    /* no, call file? */
{
    sprintf( fnam, "%s\\%s\\%s\\call\\%s",
            gisdbase, mapset, location, layer2add);
    if ( ( fp = fopen( fnam , "r" ) ) != NULL )
        laytype = 'C';
}
if ( laytype == ' ' )
    /* no, doesn't exist */
{
    SPRINTF("add_layer: no such layer %s", layer2add);
    strcpy( layer2add , "" );
    NEWVALS();
    return( -1 );
}
fclose( fp );

if ( ! del_layer_in_progress )
{
    if ( num_layers >= MAX_LAYERS ) /* add it to displayed-layer list */
    {
        SPRINTF("add_layer: screen list is full");
        strcpy( layer2add , "" );
        NEWVALS();
        return( -1 );
    }
    strcpy( dinam[num_layers] , layer2add );
    num_layers++;
}

build_legend();
if ( legend_drawn )
{
    hide_legend();
    show_legend();
}

SPRINTF( "Drawing %s" , layer2add );

if ( laytype == 'D' )
    d_digit();
else if ( laytype == 'C' )
    d_call();

strcpy( layer2add , "" );
NEWVALS();
}

del_layer ()
/* remove a layer from display */
{
    int i, idx;
    char save_ld;

    for ( i = 0 ; i < strlen( layer2del ) ; i++ ) /* assure name is uppercase */
        if ( ( layer2del[i] >= 'a' ) && ( layer2del[i] <= 'z' ) )
            layer2del[i] -= 040;

    if ( strcmp( layer2add , "DRA_MLEV" ) == 0 ) /* special case */
        dra_mlev_drawn = NO;

    /* first remove it from displayed-layer list */

    idx = -1;
    for ( i = 0 ; i < num_layers ; i++ )
        if ( strcmp( dinam[i] , layer2del ) == 0 )
            /* find it in list */

```

```

        idx = 1;
    if ( idx == -1 )
    {
        SLOUTP("layer %s is not on display", layer2del);
        strcpy( layer2del , "" );
        MWVALS();
        return( -1 );
    }

    if ( idx < num_layers - 1 )          /* if it wasn't last in list */
    {
        for ( i = idx ; i < num_layers - 1 ; i++ )    /* all move up */
            strcpy( dname[idx] , dname[idx+1] );
        strcpy( dname[num_layers-1] , "" );           /* erase last */
    }

    num_layers--;

    MWVALS();                                     /* ... gratuitous */

    /*
     * Actually delete the layer from the graphics display.
     *
     * For now, brute force: erase the graphics screen and
     * repaint all layers but the deleted one.
     */

    save_id = legend_drawn;    /* remember legend_drawn */
    erase_graphics_display();

    del_layer_in_progress = YES;

    for ( i = 0 ; i < num_layers ; i++ )
    {
        strcpy( layer2add , dname[i] );
        if ( add_layer() == -1 )
        {
            SLOUTP("?del_layer: redraw error on %s" , layer2add );
            strcpy( layer2del , "" );
            MWVALS();
            del_layer_in_progress = NO;
            return( -1 );
        }
    }

    legend_drawn = save_id;          /* recall legend_drawn */
    if ( legend_drawn )              /* & redraw it if necessary */
    {
        hide_legend();
        show_legend();
    }

    del_layer_in_progress = NO;

    strcpy( layer2del , "" );
    MWVALS();
}

store_screen ()
{
    /* if screen is blank, give up */

    if ( num_layers == 0 )
    {
        SLOUTP("?store_screen: screen is empty -- nothing to save");
        strcpy( layersavename , "" );
        MWVALS();
        return( -1 );
    }

    /* if layer with this name already exists, give up */

    sprintf( fname , "%s\\%s\\%s\\digit\\%s" ,
        gisdbase, mapset, location, layersavename );
    if ( ( fp = fopen( fname , "r" ) ) != NULL )
    {
        SLOUTP("?store_screen: layer %s already exists", layersavename);
        strcpy( layersavename , "" );
        MWVALS();
        return( -1 );
    }
}

```

```

    }
    sprintf( fname, "%s\\%s\\%s\\%s",
             gisdbase, mapset, location, layersaveame);
    if ( ( fp = fopen( fname, "r" ) ) != NULL )
    {
        SLOUTP("store_screen: layer %s already exists", layersaveame);
        strcpy( layersaveame, "" );
        SHWVALS();
        return( -1 );
    }

    laytype = 'C';

    screenfcell();
    add_to_layertxt();

    strcpy( layersaveame, "" );
    SHWVALS();
}

show_dist ()
{
    SLOUTP( "This feature is not yet available" );
}

show_coords ()
{
    SLOUTP( "This feature is not yet available" );
}

edit_colors ()
{
    SLOUTP( "This feature is not yet available" );
}

clip_to_window ()
{
    SetADR( W_l , W_b , W_r , W_t );
}

clip_to_screen ()
{
    SetADR( SCR_L , SCR_B , SCR_R , SCR_T );
}

draw_dma_elev ()                /* special branch of add_layer() */
{
    int i;

    if ( ! del_layer_in_progress )
    {
        if ( num_layers >= MAX_PLAYERS ) /* add it to displayed-layer list */
        {
            SLOUTP("add_layer: screen list is full");
            strcpy( layer2add, "" );
            SHWVALS();
            return( -1 );
        }
        strcpy( dlines[num_layers] , layer2add );
        num_layers++;
    }

    build_dma_elev_legend();      /* special legend too */

    DMA_elev_being_drawn = YES;

    if ( strcmp( vname , "Sells.C" ) == 0 )
    {
        strcpy( layer2add , "AJO_E.C" );
        SLOUTP( "Drawing AJO_E.C (DMA_ELEV part 1 of 5)" );
        d_cell();
        strcpy( layer2add , "AJO_W.C" );
        SLOUTP( "Drawing AJO_W.C (DMA_ELEV part 2 of 5)" );
        d_cell();
        strcpy( layer2add , "LOKEV.C" );
        SLOUTP( "Drawing LOKEV.C (DMA_ELEV part 3 of 5)" );
    }
}

```

```

        d_cell();
        strcpy( layer2add, "MOGAL.C" );
        SLOUT( "Drawing MOGAL.C (DMA_ELEV part 4 of 5)" );
        d_cell();
        strcpy( layer2add, "TUCSW.C" );
        SLOUT( "Drawing TUCSW.C (DMA_ELEV part 5 of 5)" );
        d_cell();
    }
    else if ( strcmp( vname, "Ajo.M" ) == 0 )
    {
        strcpy( layer2add, "AJO_E.MA" );
        SLOUT( "Drawing AJO_E.MA (DMA_ELEV part 1 of 1)" );
        d_cell();
    }
    else if ( strcmp( vname, "Sells.M" ) == 0 )
    {
        strcpy( layer2add, "AJO_E.MS" );
        SLOUT( "Drawing AJO_E.MS (DMA_ELEV part 1 of 4)" );
        d_cell();
        strcpy( layer2add, "LUKEV.MS" );
        SLOUT( "Drawing LUKEV.MS (DMA_ELEV part 1 of 4)" );
        d_cell();
        strcpy( layer2add, "MOGAL.MS" );
        SLOUT( "Drawing MOGAL.MS (DMA_ELEV part 1 of 4)" );
        d_cell();
        strcpy( layer2add, "TUCSW.MS" );
        SLOUT( "Drawing TUCSW.MS (DMA_ELEV part 1 of 4)" );
        d_cell();
    }
    else if ( strcmp( vname, "Ajo.F" ) == 0 )
    {
        strcpy( layer2add, "AJO_E.FA" );
        SLOUT( "Drawing AJO_E.FA (DMA_ELEV part 1 of 1)" );
        d_cell();
    }
    else if ( strcmp( vname, "Sells.F" ) == 0 )
    {
        strcpy( layer2add, "MOGAL.FS" );
        SLOUT( "Drawing MOGAL.FS (DMA_ELEV part 1 of 1)" );
        d_cell();
    }
    else
    {
        SLOUT( "? no DMA elevation data for this view" );
        DMA_elev_being_drawn = NO;
        DMA_elev_drawn = NO;
        return( -1 );
    }

    strcpy( layer2add, "" );
    NEWVALS();

    DMA_elev_being_drawn = NO;
    DMA_elev_drawn = YES;
}

dround ( num )          /* double-precision-to-integer rounding routine */
double num;
{
    if ( num > 0.0 ) return (int) (num + 0.499);
    else             return (int) (num - 0.499);
}

/* end of grass.c */
/*****
 *
 * ig.c -- device driver functions for Imagraph AGC hardware.
 *
 *****/

/* #define GRAPHICS_DEBUGGING 1 */

#include "stdio.h"
#include "acrtu.h"
#include "imagraph.h"

int GDMA = 0, GDSF = 0; /* debug switches */

/* Imagraph hardware/software stuff */

```

```

int    max_intensity;      /* max color intensity for this hardware */
char   *palette;           /* intensities of 1st 16 colors */
char   *model;             /* handy pointer to IG model ident */
int fillpat[16] = {-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0};
                                     /* fill pattern mask */

/* Stuff defined in Imagraph libraries */

extern int    FourBit;      /* 4 => 4-bit, 0 => 8-bit */
extern IMstru *pIMA;
extern IMstru IMmodel[];
extern SCREDEF BaseSC;
extern SCREDEF UpperSC;
extern SCREDEF LowerSC;

/* functions */

void  SCalc();
IMstru *IMOpen();
char  *palalloc();
char  *getenv();

void begin_IG_graphics ()
{
    IMstru *board;
    int    xm, ym;

#ifdef GRAPHICS_DEBUGGING
    GDBA = 0;  GDBP = 0;
    printf("\nannounce every hardware call? (1=yes,0=no): ");
    scanf("%d", &GDBA);
    if ( GDBA )
    {
        printf("\npauses after hardware call announcements? (1=yes,0=no): ");
        scanf("%d", &GDBP);
    }
#endif /* GRAPHICS_DEBUGGING */

    model = getenv("IMMODEL");
    if ( ! ( board = IMOpen(model) ) )
    {
        debugout("\nIMOpen failed: Imagraph hardware not found\n");
        exit(0);
    }
    tweakit();                /* let's tweak hardware */
    xm = board->xmapix;
    ym = board->ymapix;
    UpperSC.maxx = 0;         /* don't use upper screen */
    UpperSC.clipx = 0;
    UpperSC.maxy = 0;
    UpperSC.clipy = 0;
    LowerSC.maxx = 0;         /* don't use lower screen */
    LowerSC.clipx = 0;
    LowerSC.maxy = 0;
    LowerSC.clipy = 0;
    BaseSC.maxx = xm + 1;     /* base screen for graphics */
    BaseSC.clipx = xm;
    BaseSC.maxy = ym + 1;
    BaseSC.clipy = ym;
    SCalc(RESET_SCREEN, &BaseSC); /* reset frame buffer pointer */
    SCalc(RASE_SCREEN, &BaseSC);
    IG_hdw_init();            /* initialize hardware */
    SetOrg( BaseSC.o1, BaseSC.o2); /* coord system on base screen */
    max_intensity = (FourBit) ? 15 : 255; /* max color intensity value */
    palette = palalloc();     /* allocate palette image memory */
    load_default_palette();    /* load default palette */
    clrscrn( &BaseSC, 0 );    /* clear graphics screen to BLACK */
    SetFG( 7 );               /* default foreground is WHITE */
    SetBG( 0 );               /* default background is BLACK */
    SetPatt( -0 );            /* default pattern is solid */
}

void end_IG_graphics ()
{
    IG_hdw_release();         /* clean up hardware */
    free(palette);            /* clean up heap */
}

```

```

static IS_hdw_init ()          /* init hardware */
{
    DSInit();                  /* initialize the hardware */
    SetADR( 0, 0, pDMA->xmarpix, pDMA->ymarpix );
    initCI( 0, 0, 0, 0 );
    itstmode( 1 );              /* get us into graphics for SGA */

    return(0);
}

static IS_hdw_release ()       /* close hardware */
{
    itstmode( 0 );              /* get us out of graphics for SGA */

    return(0);
}

load_default_palette ()
{
    make_default_palette();
    putlut( Palette );

    return(0);
}

static make_default_palette ()
{
    int mi, mi34, mi2;

    mi = (FourBit) ? 15 : 255;
    mi34 = (mi >> 2) * 3;      /* 75% intensity */
    mi2 = mi >> 1;            /* 50% intensity */

    pinitCI(Palette, 0, 0, 0, 0);
    pinitCI(Palette, 1, mi, 0, 0);
    pinitCI(Palette, 2, mi, 0, mi);
    pinitCI(Palette, 3, 0, 0, mi);
    pinitCI(Palette, 4, 0, mi, mi);
    pinitCI(Palette, 5, 0, mi, 0);
    pinitCI(Palette, 6, mi, mi, 0);
    pinitCI(Palette, 7, mi, mi, mi);
    pinitCI(Palette, 8, mi2, mi2, mi2);
    pinitCI(Palette, 9, mi34, 0, 0);
    pinitCI(Palette, 10, mi, 0, mi34);
    pinitCI(Palette, 11, 0, 0, mi34);
    pinitCI(Palette, 12, 0, mi, mi34);
    pinitCI(Palette, 13, 0, mi34, 0);
    pinitCI(Palette, 14, mi34, 0, mi34);
    pinitCI(Palette, 15, mi34, mi34, mi34);

    return(0);
}

/* end of ig.c */
/*****
 *
 *      leg.c -- ASAN GRASS legend and category file handling.
 *
 *****/

#define DS(str)

#include "grass.h"
#include "globals.h"

#define LEG_FRAME_SPACE      ( charwidth / 4 )
#define LEG_FV_WIDTH        ( charwidth )
#define LEG_FV_HEIGHT        ( chardepth - 2 )
#define LEG_FVTEXT_SPACE    ( charwidth / 2 )

extern int      chardepth, charwidth;

static int      leg_frame_top;
static int      leg_frame_bot;
static int      leg_frame_left;
static int      leg_frame_right;
static int      leg_frame_height;
static int      leg_frame_width;

```

```

static char    fasm[80];
static FILE    *fp;
static int     count;
static int     fline;
static char    junk[100];

build_legend ()    /* build legend data structure */
{
    int dldx, i, j;

    num_llines = 0;

    for ( dldx = 0 ; dldx < num_layers ; dldx++ )
    {
        sprintf( fasm, "%s\\%s\\%s\\cats\\%s",
            gisdbase, mapset, location, dldnam[dldx]);
        if ( ( fp = fopen( fasm , "r" ) ) == NULL )
        {
            ERROR("build_leg: fopen %s failed", fasm);
            return( -1 );
        }

        fline = 1;

        if ( fscanf( fp , " %d %s " , &count , &fline ) < 2 )    /* read 1st line */
        {
            ERROR( "bad count or early EOF" );
            return( -1 );
        }

        for ( i = 0 ; i < 2 ; i++ )    /* skip next 2 lines */
        {
            fline++;
            if ( fgets( junk , 100 , fp ) == NULL )
            {
                ERROR( "fgets error" );
                return( -1 );
            }
        }

        for ( i = 0 ; i < count ; i++ )    /* the rest are cats */
        {
            if ( num_llines >= MAX_CATS )    /* prevent overflow */
            {
                ERROR( "legend data space full" );
                return( -1 );
            }

            fline++;
            if ( fgets( log[num_llines].str , 100 , fp ) == NULL )
            {
                ERROR( "bad data or early EOF" );
                return( -1 );
            }

            /* replace <ret> in string with <null> */

            for ( j = 0 ; j < strlen( log[num_llines].str ) ; j++ )
                if ( log[num_llines].str[j] == '\n' )
                    log[num_llines].str[j] = '\0';

            log[num_llines].pv = next_out_pv;    /* store pival */

            next_out_pv++;
            num_llines++;
        }

        fclose( fp );
    }

    DB( printf("count=%d num_llines=%d:\n", count, num_llines); )
    DB( for ( i = 0 ; i < num_llines ; i++ ) )
    DB( printf( "%s\n" , log[i].str ); )
    DB( getch(); )
}

static ERROR ( a , b , c , d , e , f , g )
{
    char msgbuf[80];
    char tmpbuf[80];

```

```

sprintf( tmpbuf , a , b , c , d , e , f , g );
sprintf( msgbuf , "build leg %s (%d): " , fnum , fline );
strcat( msgbuf , tmpbuf );
SLOUTF( msgbuf );
fclose( fp );
}

show_legend ()          /* draw legend on graphics screen */
{
    int i, j, wid, y;

    if ( legend_drawn )          /* if legend is visible, erase it */
        hide_legend();

    leg_frame_top = Wb - 4;
    leg_frame_left = 5;

    wid = 0;                      /* find widest legend line string */
    for ( i = 0 ; i < num_llines ; i++ )
        if ( ( j = strlen( leg[i].str ) ) > wid )
            wid = j;

    leg_frame_height = ( num_llines * chardepth ) + ( LEG_FRAME_SPACE * 2 );
    leg_frame_width = LEG_FV_WIDTH + LEG_FVTEXT_SPACE
        + ( wid * charwidth ) + ( LEG_FRAME_SPACE * 2 );
    leg_frame_right = leg_frame_left + leg_frame_width;

    if ( ( leg_frame_bot = leg_frame_top - leg_frame_height ) < 0 )
    {
        SLOUTF( "?legend frame below screen bottom" );
        leg_frame_bot = 1;
    }

    SetFG( C_LEG_FRAME );          /* draw frame */
    aRect( leg_frame_left , leg_frame_bot ,
        leg_frame_right , leg_frame_top , 0x40 );

    y = leg_frame_top - LEG_FRAME_SPACE - chardepth;  /* draw legend lines */
    for ( i = 0 ; i < num_llines ; i++ )
    {
        if ( y < 0 )
        {
            SLOUTF( "?legend text below screen bottom" );
            legend_drawn = YES;
            return( -1 );
        }

        SetFG( leg[i].pv );          /* draw color block */
        aRect( leg_frame_left + LEG_FRAME_SPACE , y ,
            leg_frame_left + LEG_FRAME_SPACE + LEG_FV_WIDTH , y + LEG_FV_HEIGHT ,
            0x40 );

        strcpy( txt , leg[i].str );          /* draw string */
        ix1 = leg_frame_left + LEG_FRAME_SPACE + LEG_FV_WIDTH + LEG_FVTEXT_SPACE;
        iy1 = y + ( chardepth / 2 );

        setFG( C_LEG_TEXT );
        d_text( SCREEN_COORDS );

        y -= chardepth;
    }

    if ( DRA_alav_drawn )
        show_DRA_alav_legend();

    legend_drawn = YES;
}

hide_legend ()          /* erase legend from graphics screen */
{
    if ( legend_drawn )
    {
        SetFG( C_BKG );          /* IS THIS THE CORRECT CMD?? */
        aRect( leg_frame_left , leg_frame_bot ,
            leg_frame_right , leg_frame_top , 0x40 );
    }
}

```



```

        if ( DMA_elev_drawn )
            hide_DMA_elev_legend();
    }

    legend_drawn = NO;
}

build_dma_elev_legend ()      /* build special legend for DMA elevations */
{
    /* ADD ME LATER */
}

show_DMA_elev_legend ()      /* draw DMA elevation legend */
{
    /* ADD ME LATER */
}

hide_DMA_elev_legend ()      /* erase DMA elevation legend */
{
    /* ADD ME LATER */
}

/* end of leg.c */

/*****
 *
 * It.c -- ASAM GRASS map layer list/description text file management.
 *
 * NOTE: Name "layers.txt" is wired into this code.
 *
 *****/

#include "grass.h"
#include "glocale.h"

static char    fnam[80];
static FILE    *fp;
static char    nambuf[12];

add_to_layertxt ()            /* add new layer name to the file */
{
    int i;

    if ( ( fp = fopen( "layers.txt" , "r" ) ) == NULL ) /* first find it */
    {
        perror( "fadd21: fopen[r] layers.txt failed" );
        return( -1 );
    }

    fclose( fp );

    if ( ( fp = fopen( "layers.txt" , "a" ) ) == NULL ) /* now really open it */
    {
        perror( "fadd21: fopen[a] layers.txt failed" );
        return( -1 );
    }

    /* assure layer name is uppercase */

    for ( i = 0 ; i < strlen( layersavename ) ; i++ )
        if ( ( layersavename[i] >= 'a' ) && ( layersavename[i] <= 'z' ) )
            nambuf[i] = layersavename[i] - 040;
        else
            nambuf[i] = layersavename[i];
    nambuf[i+1] = '\0';

    sprintf( fp , "%s\n" , nambuf );

    fclose( fp );
}

rm_from_layertxt ()          /* remove a layer name from the file */
{
}

```

```

/* end of lt.c */
/* ADD BETTER FIELD CHECKING. SAVE WIN & VP, CHECK AGAINST NEW ONES */

/*****
 *
 * vw.c -- read window & viewport files, set up display parameters
 *
 * FILE FORMATS ARE HARDWIRED INTO THIS CODE,
 * AND ORDER OF ITEMS IN FILES IS CRUCIAL.
 *
 *****/

#define DB(str)

#include "grass.h"
#include "globals.h"

#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <io.h>

static char fnam[80];
static char junk[20];
static FILE *fp;

set_display_params () /* set up display parameters */
{
    double h_temp, v_temp;

    /* read the window file */

    sprintf( fnam, "%s\\%s", gisbase, winname);

    if ( ( fp = fopen( fnam, "r" ) ) == NULL )
    {
        SLOUTP("vw: fopen window %s failed", winname);
        return( -1 );
    }

    fscanf( fp, "%s %d", junk, &W_l );
    fscanf( fp, "%s %d", junk, &W_r );
    fscanf( fp, "%s %d", junk, &W_b );
    fscanf( fp, "%s %d", junk, &W_t );

    fclose( fp);

    /* Find number of rows & columns in screen window */

    W_rows = W_t - W_b + 1;
    W_cols = W_r - W_l + 1;

    DB(SLOUTP("vw: W_t=%d W_b=%d W_l=%d W_r=%d W_rows=%d W_cols=%d",
              W_t,W_b,W_l,W_r,W_rows,W_cols));

    if ( ! window_frame_drawn )
    {
        SetFG( C_WIN_FRAME );
        aRect( W_l - 1, W_b - 1, W_r + 1, W_t + 1, 0x40 );
        window_frame_drawn = YES;
    }

    /* read the viewport file */

    sprintf( fnam, "%s\\%s\\%s\\viewport\\%s",
              gisbase, mapset, location, vname);

    if ( ( fp = fopen( fnam, "r" ) ) == NULL )
    {
        SLOUTP("vw: fopen viewport %s failed", vname);
        return( -1 );
    }

    fscanf( fp, "%s %d", junk, &v_proj );
    fscanf( fp, "%s %d", junk, &v_scae );
    fscanf( fp, "%s %d", junk, &v_north );
    fscanf( fp, "%s %d", junk, &v_south );
    fscanf( fp, "%s %d", junk, &v_west );
    fscanf( fp, "%s %d", junk, &v_east );
    fscanf( fp, "%s %s %d", junk, junk, &v_nw_res );
    fscanf( fp, "%s %s %d", junk, junk, &v_sw_res );

```

```

fscanf( fp , "%s %d" , junk ,      &v_format );

fclose( fp );

/*
 * Find resolution (UMs per pixel).
 *
 * To assure that maps fit in the window in both directions,
 * compute it twice (using horizontal and vertical dimensions
 * and store whichever is numerically larger.
 */

v_temp = ( v_north - v_south ) / (double)( w_t - w_b );
h_temp = ( v_east - v_west ) / (double)( w_r - w_l );

if ( v_temp >= h_temp )
    UTM_per_pixel = v_temp;
else
    UTM_per_pixel = h_temp;

DB(SPRINTF("vw: UTM_per_pixel=%f",UTM_per_pixel));

return( 0 );
}

/* end of vw.c */

```